

Series Editor

W. Bruce Croft

Editorial Board

ChengXiang Zhai

Maarten de Rijke

Nicholas J. Belkin

Charles Clarke

Giovanni Maria Sacco • Yannis Tzitzikas
Editors

Dynamic Taxonomies and Faceted Search

Theory, Practice, and Experience

 Springer

Editors

Giovanni Maria Sacco
Dipartimento di Informatica
Università di Torino
Corso Svizzera 185
10149 Torino, Italy
sacco@di.unito.it

Yannis Tzitzikas
Computer Science Department
University of Crete
P.O. Box 2208
714 09 Heraklion, Crete, Greece
tzitzik@ics.forth.gr

ISSN 1387-5264 The Information Retrieval Series
ISBN 978-3-642-02358-3 e-ISBN 978-3-642-02359-0
DOI 10.1007/978-3-642-02359-0
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2009934051

ACM Computing Classification (1998): H.3, H.5

©Springer-Verlag Berlin Heidelberg 2009

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: KünkelLopka, Heidelberg

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*A Maria Teresa e Giacomo
Giovanni Maria Sacco*

*To Tonia, my brother and parents
Yannis Tzitzikas*

Preface

Current search paradigms for the Web, direct access through search engines and navigational access via static taxonomies, have recently been strongly criticized. A third paradigm, dynamic taxonomies or faceted search, is gaining acceptance to the extent that it is now the de facto standard in product selection for e-commerce.

This new paradigm is based on a simple and easily understood visual environment which supports both direct access and guided exploration of complex information bases. While focusing on structured, guided exploration, it also bridges the gap between traditional querying and browsing. In general, query services are either too simplistic (e.g. free text queries in IR systems or Web search engines), or too complex for casual users (e.g. SQL queries, or Semantic Web queries). Browsing as well, is either too simplistic (e.g. “plain” Web links) or application specific (dynamic pages derived by specific application programs), and does not support conceptual exploration.

Dynamic taxonomies work on multidimensional taxonomies (usually organized by facets) and provide a single, coherent visual framework in which users can focus on one or more concepts in the taxonomy, and immediately see a conceptual summary of their focus, in the form of a reduced taxonomy derived from the original one by pruning unrelated concepts. Concepts in the reduced taxonomy can be used to set additional, dependent foci and users iterate in a guided yet unconstrained way until they reach a result set sufficiently small for manual inspection.

The access paradigm supported is a conceptual exploration, far more frequent in “search” tasks than the retrieval by exact specification supported by search engines and database queries. The underlying model is simple and easily understood by users, offers substantial benefits over traditional approaches, and has an extremely wide application range, and a potential for important extensions. Dynamic taxonomy/faceted search is a heavily interdisciplinary area, where data modeling, human factors, logic, inference, and efficient implementations must be considered holistically.

The goal of this book is to provide a complete and clear guide to all of the relevant aspects of dynamic taxonomies and faceted search. These include modeling, user interaction, taxonomy design, system implementation, and performance. The primary audience for this book are university students, professionals, and researchers in computer science and computer engineering who are interested in understanding and applying dynamic taxonomies, possibly in combination with other access methods, in real environments. The book may be of interest also to university students, professionals and researchers in Library and Information Science.

The book is organized as follows. Chapter 1 introduces dynamic taxonomies and faceted search. Chapter 2 formally describes faceted taxonomy-based sources. In

Chap. 3, dynamic taxonomies and faceted search are compared to other techniques including information retrieval, OLAP, dynamic result clustering, static taxonomies, decision trees, formal concept analysis, description logics, and the Semantic Web. Chapter 4 is devoted to user interface design and issues, and includes an analysis of information presentation, interaction modes, user interface design patterns, and personalized faceted search. Chapter 5 introduces important extensions to the base model, addressing data mining, structured documents, and extended expressivity through logical information systems. Chapter 6 discusses several engineering aspects of taxonomy-based sources including the integration of different and distributed taxonomy-based sources. Chapter 7 describes guidelines for schema design and the automatic construction of dynamic taxonomy schemata from textual information sources. System implementation issues are discussed in Chap. 8. Chapter 9 analyzes current and emerging application areas, including e-commerce, multimedia, e-government, human resource management and diagnostic systems. Finally, Chap. 10 synthesizes and identifies challenges and directions for further research.

Color reproductions for selected figures are included at the end of the book. These figures are referenced with a trailing C, e.g. 9.15C.

The following table cross-references some relevant topics that readers might be familiar with.

Aspect/Topic	Chapters and Sections
Real-World Applications	Chap. 8 (8.3.1), 9
Formal Concept Analysis (FCA)	Chap. 3 (3.4)
Text-Based Information Retrieval	Chap. 3 (3.1.2), 7 (7.2), 8 (8.3.1)
Web Searching	Chap. 3 (3.1.2), 8 (8.3.1)
Logics and Semantic Web	Chap. 3 (3.5), 5 (5.4, 5.5), 8 (8.5, 8.4)
Databases and OLAP	Chap. 3 (3.1), 7 (7.1.1.2), 8 (8.2)
(Graphical) User Interfaces	Chap. 4, 9
Mathematical Foundations	Chap. 2, 3, 5, 6, 8 (8.5)
Facet Analysis	Chap. 2 (2.3), 7 (7.1)

Additional and accompanying material is (and will be) available through the editors' web pages.¹

Torino (Italy)
Heraklion (Greece)

Giovanni Maria Sacco
Yannis Tzitzikas

¹<http://www.di.unito.it/~sacco/dt-book>, <http://www.ics.forth.gr/~tzitzik/fbook/>.

Acknowledgements

The editors would like to thank all the authors for their contributions and their effort in making this book possible. Special thanks to Sébastien Ferré and to Saverio Perugini for their extensive effort in reviewing early versions of the entire book, to Anastasia Analyti and to Giorgos Flouris for proof reading some chapters, and to Stella Kopidaki for checking and correcting the bibliographic references.

Giovanni Maria Sacco would like to thank Antonio Di Leva and Piercarlo Giolito for their helpful comments.

Yannis Tzitzikas would like to thank his collaborators including Nicolas Spyrtos, Anastasia Analyti, Panos Constantopoulos and Carlo Meghini, as well as his students Nikos Armenatzoglou, Stella Kopidaki, Myron Papadakis, and Panagiotis Papadakos for their contributions and for proof reading several parts of the book.

Sébastien Ferré wishes to thank the members of the research group LIS, who are implied in some of the ideas that are presented in this book. Special thanks to Olivier Ridoux, who is at the origin of the Logical Information Systems, and also to the several PhD students who contributed to freshen and extend them in many directions: Yoann Padioleau, Benjamin Sigonneau, Peggy Cellier, Olivier Bedel, and Pierre Allard.

Contents

1	The Model	1
	Giovanni Maria Sacco	
1.1	Exploratory Search	3
1.2	Dynamic Taxonomies Defined	4
1.2.1	Concepts	6
1.2.2	Relationships Among Concepts	7
1.2.3	Reduced Taxonomies and Exploration	8
1.3	Implications for Schema Design	11
1.4	Advantages	13
1.5	Application Areas	14
1.6	Faceted Search and Dynamic Taxonomies	15
1.7	Book Roadmap	16
2	Faceted Taxonomy-Based Sources	19
	Yannis Tzitzikas	
2.1	Introduction	19
2.2	Taxonomies and Partially-Ordered Sets	20
2.3	Faceted Taxonomies	21
2.4	Taxonomy-Based Sources	24
2.5	On Intervals	26
2.6	Modeling Interaction	28
2.6.1	Zoom Points	29
2.6.2	Zoom-in	30
2.6.3	Zoom-Side	31
2.6.4	Zoom-out	32
2.7	Restriction	33
3	Comparison with Other Techniques	35
	Giovanni Maria Sacco, Sébastien Ferré, and Yannis Tzitzikas	
3.1	Structured Access and Information Retrieval	35
3.1.1	Queries on Structured Data	38
3.1.2	Information Retrieval	41
3.2	Static Taxonomies	46
3.2.1	Static Monodimensional Taxonomies	48
3.2.2	Static Multidimensional Taxonomies with no Concept Composition Capabilities	48
3.2.3	Multidimensional Taxonomies with Concept Composition Capabilities	49

- 3.2.4 A Comparison 51
- 3.2.5 Taxonomy Pruning in Dynamic Taxonomies 52
- 3.2.6 Relaxing the Assumptions 54
- 3.2.7 Experimental Results 55
- 3.3 Decision Trees 57
- 3.4 Formal Concept Analysis 59
 - 3.4.1 Data Model 59
 - 3.4.2 Information Access 62
 - 3.4.3 Conclusion 64
- 3.5 Semantic Web 64
 - 3.5.1 Description Logics and OWL Ontologies 66
 - 3.5.2 Semantic Web and Exploratory Search 70
 - 3.5.3 Conclusions 72
- 4 User Interface Design 75**

Moritz Stefaner, Sébastien Ferré, Saverio Perugini, Jonathan Koren, and Yi Zhang

 - 4.1 Principles 75
 - 4.2 Challenges 77
 - 4.3 Navigation Modes 78
 - 4.3.1 Zoom-in 81
 - 4.3.2 Zoom-out 82
 - 4.3.3 Shift 82
 - 4.3.4 Pivot 83
 - 4.3.5 Slice and Dice 84
 - 4.3.6 Range Selection 85
 - 4.3.7 Querying by Examples 85
 - 4.4 Design Patterns 86
 - 4.4.1 Selection Management 87
 - 4.4.2 Revealing Hierarchy 88
 - 4.4.3 Facet Management 90
 - 4.4.4 Keyword Search 91
 - 4.4.5 Filter Summary and History Navigation 93
 - 4.4.6 Animated Transitions 93
 - 4.4.7 Visualizing Proportions 94
 - 4.5 Extensions and Related Approaches 95
 - 4.5.1 FaThumb 96
 - 4.5.2 Browsing Related Entities 97
 - 4.5.3 Resource Analytics 98
 - 4.5.4 Out-of-turn Interaction 99
 - 4.6 Personalizing Faceted Search 102
 - 4.6.1 Introduction 102
 - 4.6.2 Related Work: Personalized Search and Filtering 103
 - 4.6.3 Personalization Based on Collaborative Filtering 104
 - 4.6.4 K-Nearest Neighbors Based on Item–Item Similarity or User–User Similarity 105

- 4.6.5 Singular Value Decomposition 106
- 4.6.6 Personalization Using Content Based Filtering 107
- 4.6.7 An Ontological Approach 109
- 4.6.8 Evaluation Regime 110
- 4.6.9 Conclusions 112

- 5 Extensions to the Model 113**
- Giovanni Maria Sacco and Sébastien Ferré
- 5.1 Data Mining 114
 - 5.1.1 Association Rule Mining 114
 - 5.1.2 Dynamic Taxonomy Foci and Association Rules 114
 - 5.1.3 Integrating Association Rules with Dynamic Taxonomies 116
 - 5.1.4 An Example 118
 - 5.1.5 Father–Son Correlation in the Taxonomy 120
 - 5.1.6 General Association Rules 120
 - 5.1.7 Side-by-Side Comparison 122
- 5.2 Structured Objects 123
- 5.3 Virtual Concepts 126
- 5.4 Logics 127
 - 5.4.1 From Taxonomies to Logics 127
 - 5.4.2 From Logics to Dynamic Taxonomies 130
- 5.5 Web Ontologies 133
 - 5.5.1 Re-defining Extensions and Dynamic Taxonomies 133
 - 5.5.2 Additional Navigation Modes 136
- 5.6 Fuzzy Dynamic Taxonomies 138
- 5.7 Miscellanea 139
 - 5.7.1 Predefined Foci for Personalization and Access Control 139
 - 5.7.2 SAES Facets 140
 - 5.7.3 Popularity, Recommendations, and Authoritativeness 142
 - 5.7.4 Augmenting IR Recall 143

- 6 Engineering Taxonomy-Based Sources 145**
- Yannis Tzitzikas
- 6.1 Compound Terms Composition Algebra (CTCA) 145
 - 6.1.1 Motivation 145
 - 6.1.2 The Algebra in Brief 146
 - 6.1.3 Deriving Navigational Trees from CTCA Expressions 154
 - 6.1.4 Tackling the Taxonomy Evolution Problem 155
 - 6.1.5 Expression Mining and Other Applications 159
- 6.2 Adaptation of Taxonomy-Based Sources Through User Feedback 161
- 6.3 Mapping Taxonomy-Based Sources 163
- 6.4 Distributed Taxonomy-Based Sources 169
 - 6.4.1 Mappings and Mediators 169
 - 6.4.2 Distributed Query Evaluation 172
- 6.5 Synopsis and Bibliographic References 174

7	Taxonomy Design	175
	Wisam Dakka, Panagiotis Ipeirotis, and Giovanni Maria Sacco	
7.1	General Guidelines for Taxonomy Design	175
7.1.1	Design ‘in the Small’	180
7.1.2	Design ‘in the Large’	186
7.2	Automatic Construction from Text Information Bases	190
7.2.1	Problem Overview	190
7.2.2	Supervised Facet Extraction for Collections of Text-Annotated Items	192
7.2.3	Unsupervised Facet Extraction for Collections of Text Documents	195
7.2.4	Evaluating Our Supervised Facet Extraction Technique	202
7.2.5	Evaluating Our Unsupervised Facet Extraction Technique	205
7.2.6	Further Discussion and Future Work	212
7.2.7	Conclusion	212
8	System Implementation	215
	Giovanni Maria Sacco, Yannis Tzitzikas, and Sébastien Ferré	
8.1	Architecture and Implementation Strategies	215
8.1.1	Logical Architecture	216
8.1.2	Computing the Focus	218
8.1.3	Computing the Reduced Taxonomy	220
8.1.4	Presentation Strategies	221
8.1.5	Physical Storage Structures for the Extension	222
8.1.6	Experimental Data	228
8.1.7	Further Performance Enhancements	234
8.2	Implementation over a Relational Database Management System	237
8.3	Case Studies: Existing Systems	241
8.3.1	FleXplorer and Mitos	245
8.3.2	FASTAXON	247
8.4	Formats and Protocols	250
8.5	Composition of Taxonomies with Logic Components	253
8.5.1	Logics	254
8.5.2	Logic Functors	256
8.5.3	Combining Attributes, Concrete Domains, and Taxonomies	256
8.5.4	Reconstructing the Description Logic \mathcal{ALC}	258
8.5.5	Conclusion	260
9	Applications and Experiences	263
	Giovanni Maria Sacco and Sébastien Ferré	
9.1	Introduction	263
9.2	E-commerce	265
9.2.1	The Thinning Game	266
9.2.2	The End Game	269
9.3	Multimedia Information Bases	272

- 9.3.1 Combining Conceptual Access with Low Level
Multimedia Features 274
- 9.3.2 Monodimensional vs. Multidimensional Clustering for
Low Level Features 275
- 9.3.3 Representing Low Level Multimedia Features 277
- 9.3.4 Examples of Exploration 279
- 9.4 Diagnostic Systems 282
 - 9.4.1 Computer-Assisted Medical Diagnosis 282
 - 9.4.2 Diagnosis Through Dynamic Taxonomies 283
 - 9.4.3 Application of Dynamic Taxonomies to the Diagnosis of
Rare Diseases 284
- 9.5 Digital Libraries and News Systems 288
- 9.6 E-government 290
- 9.7 File Systems 294
 - 9.7.1 Implementation 294
 - 9.7.2 Applications 295
 - 9.7.3 Related Works 297
- 9.8 Geographical Information Systems 298
 - 9.8.1 Data Model and Querying Language 298
 - 9.8.2 Interface and Implementation of GEOLIS 300
 - 9.8.3 Experiments 301
- 10 Conclusions 303**
Giovanni Maria Sacco and Yannis Tzitzikas
- References 307**
- Index 323**
- Appendix A Color Images 329**

List of Contributors

Wisam Dakka Google Inc., New York, USA, wisam.dakka@gmail.com

Sébastien Ferré Irisa, Université de Rennes 1, Campus de Beaulieu,
35042 Rennes cedex, France, ferre@irisa.fr

Panagiotis Ipeirotis Department of Information, Operations, and Management
Sciences, Leonard N. Stern School of Business, New York University,
44 West Fourth Street, New York, NY 10012, USA, panos@stern.nyu.edu

Jonathan Koren University of California, Santa Cruz, 1156 High Street,
Santa Cruz, CA 95064, USA, jonathan@soe.ucsc.edu

Saverio Perugini Department of Computer Science, University of Dayton,
Dayton, OH 45469-2160, USA, saverio@udayton.edu

Giovanni Maria Sacco Dipartimento di Informatica, Università di Torino, Corso
Svizzera 185, 10149 Torino, Italy, sacco@di.unito.it

Moritz Stefaner Interaction Design Lab, University of Applied Sciences
Potsdam, Pappelallee 8-9, 14469 Potsdam, Germany, moritz@stefaner.eu

Yannis Tzitzikas Computer Science Department, University of Crete,
P.O. Box 2208, 714 09 Heraklion, Crete, Greece, tzitzik@ics.forth.gr;
Institute of Computer Science, Foundation for Research and Technology—Hellas
(FORTH), N. Plastira 100, Vassilika Vouton, 700 13 Heraklion, Crete, Greece

Yi Zhang University of California, Santa Cruz, 1156 High Street, Santa Cruz,
CA 95064, USA, yiz@soe.ucsc.edu

Chapter 1

The Model

Giovanni Maria Sacco

*Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?*

T.S. Eliot, Choruses from 'The Rock', 1934

We live in a world where the quantity of available information and its rate of growth are rapidly becoming limiting factors as important as the lack of information has been for thousands of years. The Internet and the World Wide Web are the main enabling technologies for this shift. In the past few years, the global distribution of information through Internet has made an enormous mass of information available to any web-connected location in the world. The physical location of information (large libraries, museums, etc.), one of the largest limiting factors in information availability, is now immaterial. With recent advances in wireless communications, all the information is also available on-the-move.

At the same time, the conversion of existing information (books, images, etc.) to digital format and the creation of new information in the appropriate format, has proved less overwhelming than it appeared in the early 1990's. Social networking and collaborative work and the distributed gathering/conversion of information has caused the quantity of electronically available information to grow at an extremely fast rate.

This situation has resulted in a dramatic information overload. After a decade of using traditional access paradigms, such as queries on structured database systems and information retrieval or search engines, the feeling that “search does not work” and “information is too hard to find” is now reaching a consensus level.

Two different information access modes can be identified: *focalized search* vs. *exploratory search*. In focalized search, the user attempts to quickly locate relevant information items on the basis of their contents. In exploratory search (also called browsing) the user explores relationships among items in a database. For example, consider a student using an electronic encyclopedia to produce a term paper on Michelangelo [236]. He quickly locates the section on Michelangelo (among several thousand other sections). This is focalized search. At that point, he explores relationships between Michelangelo and other painters, sculptors, architects, the Italian Renaissance at large, and the political situation in Italy during that period, etc. This is exploratory search.

Traditionally, research focused on focalized search. Examples include queries on structured databases and information retrieval (IR) techniques [311], recently

dubbed search engines. Database queries require structured data and are not easily applicable to many practical situations, in which most information is unstructured or semi-structured. IR techniques try to implement querying for precise results on textual unstructured information bases.¹ Despite their wide usage, the limitations of IR techniques are well known: Blair and Maron [44] report that only 20% of relevant documents are actually retrieved. Such a significant loss of information is due to the extremely wide semantic gap between the user model (concepts) and the model used by commercial retrieval systems (words or strings of characters). Other problems include poor user interaction because the user has to formulate her query with no or very little assistance, and no exploration capabilities since results are presented as a flat list with no systematic organization.²

In order to overcome the semantic gap inherent in current IR technology, static taxonomies (such as *Yahoo!*'s) can be used. Such taxonomies are based on a hierarchy of concepts which can be used to select areas of interest and restrict the portion of the infobase to be retrieved. Taxonomies support abstraction and are easily understood by end users. Document classification according to taxonomy entries, usually a manual process although automatic and semiautomatic techniques have been proposed, eliminates the IR semantic gap because documents are now semantically organized. Taxonomies no longer deal with actual data, but rather with metadata, i.e., with a uniform, standardized and controlled description of content which is independent of format and, in the case of textual documents, of the actual terms found in the document.

However, static taxonomies are not scalable for large information bases [247], and the average number of documents retrieved becomes rapidly too large for manual inspection. The rapid decline of *Yahoo!*'s taxonomy as a primary tool to access information provides pragmatic evidence.

Solutions based on semantic networks, general ontologies, and the Semantic Web [40] are more powerful than traditional taxonomies. However, general semantic schemata are intended for programmatic access and are known to be difficult to understand and manipulate by the casual user. Therefore, user interaction must be mediated by specialized agents, and this increases costs, time to market, and decreases the transparency and flexibility of user access. In particular, agent-mediated search is based on the classic knowledge-based system paradigm, which does not take the user into account, but rather establishes a master-slave relationship between the system and the user. Brézillon [48] identifies this as the primary cause of failure for knowledge-based systems and, in particular, the fact that such systems behave as oracles, providing answers they are usually unable to explain to the user.

Regardless of approach, the underlying access paradigm is still focalized search: the user is assumed to know what he wants and the system tries to materialize the

¹The term information base or infobase, instead of database, is used here to denote a collection of heterogeneous data objects of different format (text, images, video, etc.) which is not necessarily structured.

²Recent advances, including tag clouds, additional keyword suggestions, and result clustering address part of these problems and are reviewed in Chap. 3.

result. The techniques differ mainly in the amount of intelligence the system devotes to understanding what the user wants.

1.1 Exploratory Search

We contend that most “search” tasks are exploratory and imprecise in essence, and that using a focalized search paradigm in this context leads perforce to inadequate or frustrating user interactions. There are many different reasons why a user needs to explore an information base and consequently many exploratory patterns occur in practice and often depend on the application domain.

Perhaps the most common exploratory pattern is the pragmatic *find the right object* or *object-seeking* pattern which commonly occurs in e-commerce and in other object-selection tasks. We are given an information base consisting of objects characterized by features, and a user who wants to find the object which best suits his needs. For example, consider the purchase of a digital camera. Each camera is characterized by features such as price, weight, resolution, etc. If the user’s primary goal is to minimize cost, he will strive for the best inexpensive camera, with the definition of best inexpensive camera depending on his secondary goal (e.g., high resolution). Thus, for this exploratory approach to be effective, the user needs to:

- quickly find all possibly relevant features. The number of features might be overwhelming and they should be organized in a systematic way to allow user data access at different levels of abstraction: a taxonomic organization is a requirement in most cases;
- freely focus on the most relevant feature according to his individual requirements and discard objects without that feature. In the example above, the user might select a certain price range (e.g., cameras costing less than \$200) as a starting point. Cameras outside that range must be discarded;
- explore all the features correlated with the selected one. What are the features (e.g., resolution, zoom, etc.) for cameras under \$200? If the user is unable to determine them easily, the next focus cannot be set and the user has to inspect all the inexpensive cameras and find their features by enumeration. On the other hand, if related features are available, she can add to the current focus the next feature in the order of perceived importance and focus on it, thereby discarding other cameras which do not have that feature and consequently further thinning the number of candidate objects.

Advocates of the system-centric approach adopted by agent-mediated search might contend that these requirements are artificial, because the problem can be stated as a classical optimization problem: the “best” camera is the one which minimizes a weighted combination of features. Hence, a precise search rather than user exploration can be used.

From a user-centered perspective, weights cannot be obviously predefined, even for a specific user. Different users have different perceptions, but even the same user can be cost-conscious today, and less so tomorrow after a raise in salary. Thus,

weights must be supplied each time by the user according to his perceptions. The user must supply a potentially large number of weights, which is cumbersome, and understand the effects of these weights on the selection mechanism of the agent, which is hopelessly difficult.

On the other hand, the mechanism sketched above only requires that the user identifies the most important feature: ranking instead of weighting, further simplified by the hierarchical abstraction mechanism of a taxonomy. Easy understanding of the consequences of focusing and real-time implementations allow the user to experiment with different strategies and gives the user the feeling to have considered all the possibilities.

The same pattern applies to all object-selection tasks in which different criteria must be reconciled to find a number of *good* candidates. E-commerce is an obvious application, but personnel selection, medical diagnosis, person identification, etc. fall in the same paradigm.

Not all the exploratory search patterns are so pragmatic and require the selection of objects. The exploratory search on Michelangelo discussed above, is a *knowledge-seeking* rather than an object-selection task. In other words, here we use Michelangelo as a focus to get additional relevant knowledge on a specific topic: the goal is to increase our knowledge, rather than pragmatically using this knowledge to select an object.

At the extreme end of this scale, there are *wisdom-seeking* exploratory patterns which are becoming increasingly important, in which the goal of exploration is to understand the inner laws of the information base, and gain insight into the working of the phenomena described by the information base. In these emerging applications, so to say, the journey is much more important than the destination. These exploratory patterns require data-mining capabilities.³ Some applications are discussed in Sect. 8.5.5.

The best known technique for explorative search is currently hypertext/hypermedia [124]. Hypermedia connects items in the information base through explicit links which the user can traverse at runtime according to his needs. As the vast literature on this subject shows, hypermedia is able to accommodate a large number of user exploratory access patterns. Hypermedia is quite flexible, but it gives no systematic picture of the relationships among documents because there is no systematic organization of links by abstractions, and exploration must consequently be performed one-document-at-a-time, which is quite time consuming and ineffective.

1.2 Dynamic Taxonomies Defined

Traditional access paradigms are not suited to search tasks that are exploratory and imprecise: the user needs to explore the information base, find relationships among

³See Sect. 5.1.

concepts, and thin alternatives out in a guided way. New access paradigms supporting exploration are needed. Dynamic taxonomy and faceted search systems focus on user-centered interactive exploratory access, and propose a holistic approach in which modeling, interface, and interaction issues are considered together.

One of the key factors of this model is an explicit quest for simplicity and minimality, as opposed to current research trends which tend to high-complexity solutions. The effort in reducing the model to its minimal components makes it easily understandable and usable by end-users with no need for the mediation of any agent. As importantly, it makes the model easily understandable to researchers as well, and represents a solid foundation on which several extensions and solutions to real-world problems can be built.

Dynamic taxonomies [231, 232, 234, 236] (*DT*, also recently known as *faceted search systems*) are a general knowledge management model based on a multidimensional⁴ classification of heterogeneous data *objects*⁵ and are used to explore/browse complex information bases in a guided yet unconstrained way through a visual interface. The model is primarily concerned with user-centered access, and object classification is not addressed in the base model.

The conceptual schema or intension of a dynamic taxonomy is a plain taxonomy designed by an expert of the domain: a concept hierarchy going from the most general to the most specific concepts and not requiring any other relationship in addition to *subsumptions*. A concept *A* is subsumed by a concept *B* ($A \leq B$) if the set of instances classified under *A* is intensionally constrained to be equal to or a subset of the set of instances classified under *B*: $A \subseteq B$. Subsumption models taxonomic IS-A relationships. In this case, $A \leq B$ means either that $A \equiv B$ or that *A* is a descendant of *B* in the taxonomy, so that subsumptions define a partial order among concepts. Directed acyclic graph taxonomies modeling multiple inheritance are supported.

Objects in the extension are abstract, and consequently objects of any type and format can be managed in a uniform way. Abstract objects can be freely classified under *n* ($n > 1$) concepts at any level of abstraction (i.e. at any level in the conceptual tree). This multidimensional classification is a generalization of the monodimensional classification scheme used in conventional taxonomies and models common real-life situations. First, objects are very often about different concepts: for example, the present book can be classified under “information retrieval”, “ontologies”, “multimedia databases”, etc. Second, objects to be classified usually have different features, perspectives, or facets (e.g., Time, Location, etc.), each of which can be described by an independent taxonomy. Taxonomies with a multidimensional classification will be called *multidimensional taxonomies*.

⁴The reader is referred to Gärdenfors [116] for a detailed discussion of multidimensional spaces in cognitive science and concept formation, induction and semantics.

⁵The term *object* is used to denote an abstract information item which is atomic and whose content and medium are transparent to the model. Objects have been called terms, documents, items, resources, and atoms in literature.

1.2.1 Concepts

A concept C is just an abstract label which identifies all the objects classified under C . Concepts are not textual terms. Although concepts are often externally shown as textual labels, these labels need not have any connection with terms contained in the objects they represent; incidentally, such objects are not necessarily textual. In addition, although concept labels are used to convey the concept meaning to users, such a meaning is not exploited by the model, so that for modeling purposes, each concept can be identified by a unique abstract identifier (such as a unique numeric id).

Concepts are defined by their extension rather than by specific properties they exhibit. Two different types of extension for a concept C are defined. The shallow extension of C (denoted by $shallowExtension(C)$) is defined as the set of objects directly classified under C . The deep extension of C (denoted by $deepExtension(C)$) includes all the shallow extensions for the conceptual subtree rooted in C :

$$\begin{aligned}
 & deepExtension(C) \\
 &= \{d \mid d \in shallowExtension(C') \wedge (C' = C \vee C' \text{ is a descendant of } C)\}
 \end{aligned}$$

or, equivalently

$$deepExtension(C) = \{d \mid d \in shallowExtension(C') \wedge (C' \leq C)\}$$

By construction, the shallow and the deep extension for a terminal (leaf) concept are the same. An example of each type of extension is given in Fig. 1.1. The intension is above the line, and the extension is below the line. Circles represent concepts, and objects are represented by rectangles. Solid arcs represent subsumptions, and dotted arcs represent classifications.

We claim that the ‘natural’ semantics of C is the deep extension of C , because a specific level of abstraction subsumes all its specializations. When we refer to animals we include dogs, cats, aardvarks, etc., and all the objects classified under them. Therefore $objects(C) = deepExtension(C)$. For this reason, the *extension* of

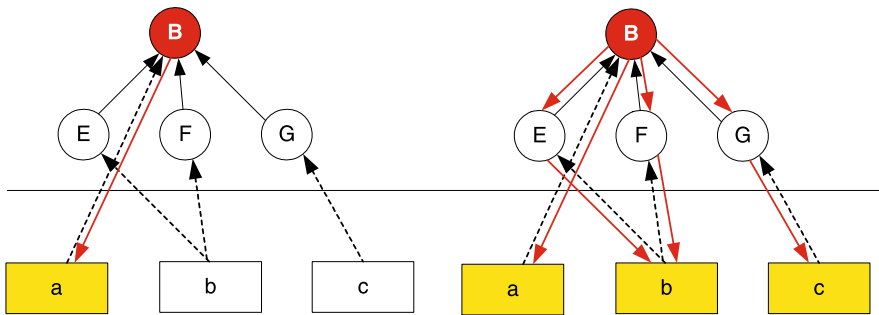


Fig. 1.1 Shallow (at left) and deep (at right) extensions for concept B

a concept C refers to the deep extension of C in the following, unless otherwise noted.

Subsumptions require that an inclusion constraint is maintained. If C' is a descendant of C in the taxonomy, then $objects(C') \subseteq objects(C)$. If shallow and deep extensions are explicitly stored, this results in a form of *backward inheritance*. An object in the shallow extension of a concept C is also classified in the deep extension of each ancestor of C . The deep extension of the root concept of the taxonomy includes the entire universe U of objects.

The shallow extension is needed only if objects can be classified at any level of the conceptual tree. In the rather common case in which objects are only classified under terminal concepts, the shallow extension is not required, because it is empty for non terminal concepts, and equivalent to the deep extension otherwise.

An immediate consequence of the interpretation of concepts as sets of objects is that logical operations on concepts can be performed by the corresponding set operations on their extensions. The information base can be manipulated and derived concepts created by combining concepts through the normal logical operations (and, or, not).

1.2.2 Relationships Among Concepts

1.2.2.1 The Base Extensional Inference Rule

In conventional taxonomies, concepts are only related by subsumptions. In a dynamic taxonomy, concept relationships other than subsumptions are inferred through the extension only, according to the following base extensional inference rule:

Two concepts A and B are related (denoted by $A \rightleftharpoons B$) iff there is at least one object d in the extension which is classified at the same time under A or under one of A 's descendants and under B or under one of B 's descendants.

For example, an *unnamed*⁶ relationship between terrorism and New York can be inferred, if an object classified under terrorism and New York exists. At the same time, since New York is a descendant of USA, also a relationship between terrorism and USA can be inferred. The extensional inference rule can be seen as a device to infer relationships on the basis of empirical evidence.

Equivalent definitions of the base extensional inference rule are:

$$A \rightleftharpoons B \quad \text{iff} \quad \exists o \in U : o \in objects(A) \wedge o \in objects(B)$$

$$A \rightleftharpoons B \quad \text{iff} \quad objects(A) \cap objects(B) \neq \emptyset$$

Subsumption relationships in the taxonomy are, by definition, a special case of the extensional inference rule.

⁶Although the name or type of relationships cannot be known through the inference rule only, they can be conveyed through specific design rules for the taxonomy as discussed in Sect. 7.1.1.

1.2.2.2 The Extensional Inference Rule

The base extensional inference rule can be extended to cover the relationship between a given concept C and a concept expressed by an arbitrary subset S of the universe: C is related to S iff there is at least one object o in S which is also in $objects(C)$ or, equivalently, iff $objects(C) \cap S \neq \emptyset$.

Hence, the extensional inference rule can infer relationships not only for base concepts, but also for any logical combination of concepts. Since it is immaterial how S is produced, dynamic taxonomies can infer relationships between a concept and sets of objects produced by other retrieval methods such as database queries, shape retrieval, etc. and, therefore, access through dynamic taxonomies can be easily combined with any other retrieval method. The extensional inference rule reduces to the base rule when the set S is the deep extension of a concept C' .

Given a taxonomy, the set of concepts related to a set S is called the *related set of concepts* ($RS(S)$) and is defined as: $RS(S) = \{C \mid objects(C) \cap S \neq \emptyset\}$. Because of the inclusion constraint in subsumption, if $C \in RS(S)$ and C' is an ancestor of C , then $C' \in RS(S)$. Conversely, if $C \notin RS(S)$ and C' is a descendant of C , then $C' \notin RS(S)$.

In interaction, it is quite useful to know for each C related to S , how many objects are in the intersection $objects(C) \cap S$. We call this quantity *related count*, $rc(C|S)$: $rc(C|S) = |objects(C) \cap S|$. By definition, $rc(C|U) = |objects(C)|$. The related set of concepts can be reformulated in terms of the related count: $RS(S) = \{C \mid rc(C|S) > 0\}$.

1.2.3 Reduced Taxonomies and Exploration

Given a set of objects S , the extensional inference rule can be used to produce a conceptual summary of S according to the original taxonomy by simply pruning from the taxonomy all those concepts C which are not related to S , i.e. all $C \notin RS(S)$. This taxonomy is called a *reduced taxonomy*, $RT(S)$.

The fundamental idea for user-centered exploration is to use the taxonomy to:

- (a) set an *interest focus* as a boolean combination of concepts or through an external query, and
- (b) summarize concepts related to the interest focus through a reduced taxonomy, from which unrelated concepts are pruned.

This means that the original taxonomy can adapt to and summarize any subset of the universe (hence the term *dynamic taxonomy*), whereas traditional static taxonomies can only summarize the entire universe.

The initial user interest focus F_0 is the universe U , i.e., all the objects in the information base. The user is initially presented with a tree representation of the initial taxonomy for the entire knowledge base. Each concept C has also a related count

$rc(C|F_i)$.⁷ The related count is a function of the current focus F_i . Since $F_0 = U$, $rc(C|F_0)$ is equivalent to $|objects(C)|$, i.e., to the cardinality of the deep extension of C .

In the simplest case, the user selects a concept C in the taxonomy and zooms over it. The *zoom operation*⁸ changes the current state in two ways. First, the current focus F_i becomes $F_{i-1} \cap objects(C)$. Objects not in the focus are discarded. Second, the tree representation of the taxonomy is modified to summarize the new focus. All and only the concepts related to F_i are retained and the count for each retained concept C' is updated to reflect the number of objects in the focus F_i which are classified under C' , i.e., $rc(C'|F_i)$. The reduced taxonomy is derived from the initial taxonomy by pruning all the concepts not related to F_i ,⁹ and it is a conceptual summary of the set of objects identified by F_i , in the same way as the original taxonomy is a conceptual summary of the universe.

The retrieval process can be seen as an iterative thinning of the information base: the user selects a focus, which restricts the information base by discarding all the objects not in the current focus. Only the concepts used to classify the objects in the focus and their ancestors are retained. These concepts, which summarize the current focus, are those and only those concepts which can be used for further refinements. From a human-computer interaction perspective, the user is effectively guided to reach his goal by a clear and consistent listing of all possible alternatives, and this interaction is often called *guided thinning* or *guided navigation*.

Figures 1.2–1.5 show how the zoom operation works. Figure 1.2 shows a dynamic taxonomy.

We assume that the user zooms on concept C . First, the interest set (i.e. the extension of the interest focus) is computed. In this case, the interest set is the deep extension of C , i.e., $deepExtension(C) = \{b, c, d\}$. This set is computed by following downwards in Fig. 1.3 all the arcs incident to C or one of its descendants $\{H, I\}$. All the objects not in the interest set can be removed from the extension, which is therefore reduced.

Then the reduced taxonomy for this interest set is computed. First, all the concepts related to the interest focus ($RS(C)$) are computed. According to the extensional inference rule, these are all the concepts under which at least an object in the interest set is classified. In Fig. 1.4, we compute this set by following all the arcs originating from the interest set. The set of concepts related to the focus is therefore $RS(C) = \{F, G, H, I, B, C, A\}$.¹⁰

⁷Related counts are generally considered very important in guiding user interactions. However, some systems do not show them, for performance reasons.

⁸Also called zoom-in afterwards.

⁹For simplicity, we assume here that pruned concepts are not shown to the user. However, the very fact that a concept was pruned might be an important information for the user. In this case, pruned concepts are shown to the user in an appropriate format which indicates that they have a related count equal to zero and are not selectable for additional zoom operations.

¹⁰As remarked before, if a concept is related to a set S , also all of its ancestors are related to S .

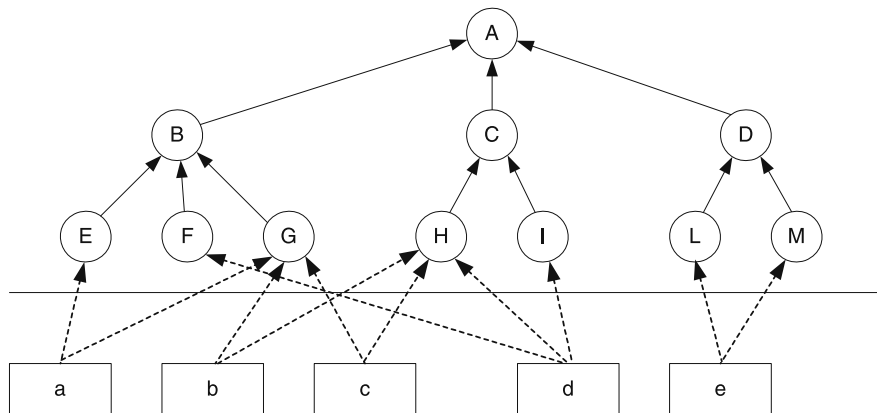


Fig. 1.2 A dynamic taxonomy: the intension is above the line, the extension below. *Solid arcs* denote subsumptions, and *dotted arcs* represent classification

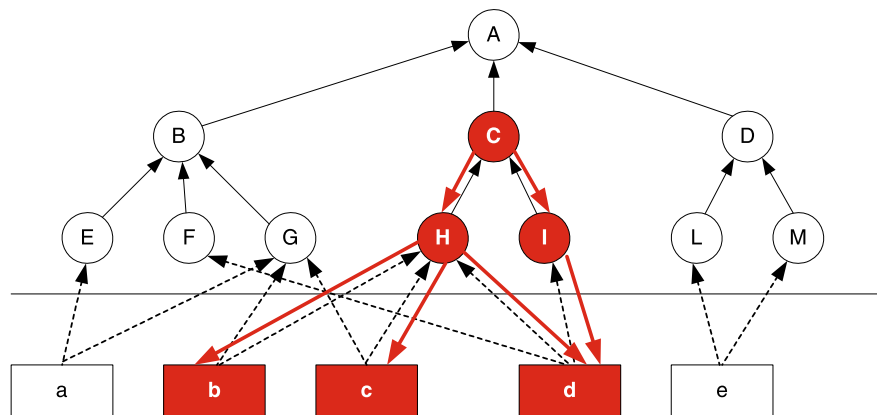


Fig. 1.3 Computing the deep extension for C

Finally, in Fig. 1.5, all the concepts not in $RS(C)$ are removed from the intension, thus producing a reduced taxonomy which fully describes all and only the objects in the current focus C.

A subsequent zoom operation can only be performed on concepts in the reduced taxonomy, i.e., on concepts in $RS(F_1)$. This guarantees that no empty results are ever found, by construction, and that, at any stage, unrelated (and therefore irrelevant) concepts are discarded. The selection of a concept C_2 from the reduced taxonomy determines a new focus $F_2 = F_1 \cap C_2$. For example, the selection of G in the context of F_1 determines $F_2 = \{b, c\}$ and $RS(F_2) = \{G, H, B, C, A\}$.

It is possible to extend zoom operations defined on a single concept to zooms defined on expressions of concepts or on the result of external query methods.

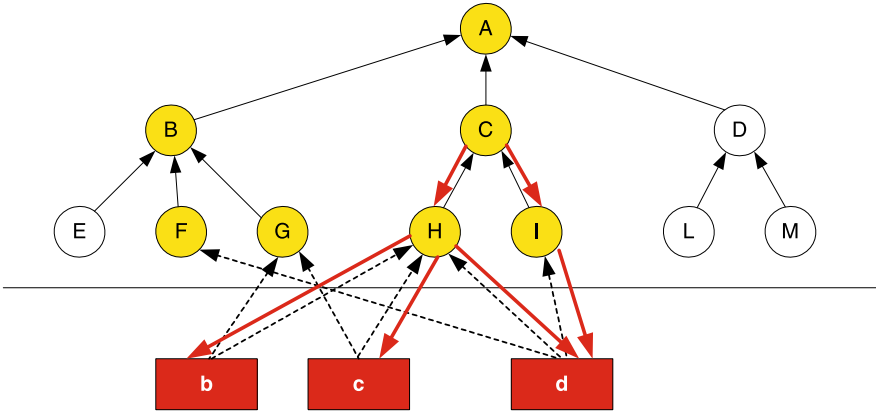


Fig. 1.4 Computing the related set for C ($RS(C)$)

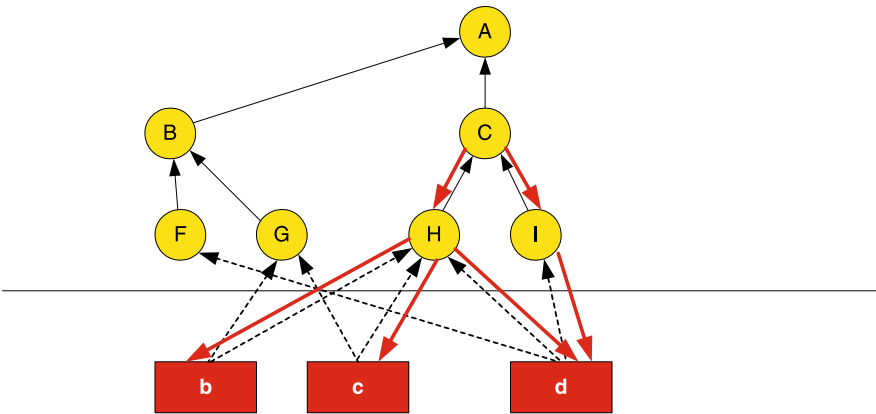


Fig. 1.5 The reduced taxonomy for focus C

1.3 Implications for Schema Design

The derivation of concept relationships through the extensional inference rule has important implications on conceptual modeling. First, a fundamental point is that relationships among concepts need not be anticipated but can be inferred from the actual classification. This simplifies taxonomy creation and maintenance. In traditional approaches, only the relationships among concepts explicitly described in the conceptual schema are available to the user for browsing and retrieval. Therefore, all possible relationships must be anticipated and described: a very difficult if not impossible task. In dynamic taxonomies, no relationships in addition to subsumptions are required, because conceptual relationships are automatically derived from the actual classification. For this reason, dynamic taxonomies easily adapt to new relationships and are able to discover new, unexpected ones.

Second, since dynamic taxonomies synthesize compound concepts, these usually do not need to be represented explicitly. This property removes the main cause of the combinatorial growth of traditional taxonomies. Sacco [236] developed guidelines which produce taxonomies that are compact and easily understood by users. Some are similar to faceted classification [134, 224], at least in its basic form: the taxonomy is organized as a set of independent, ‘orthogonal’ subtaxonomies (facets or perspectives) to be used to describe data. Additional guidelines [236] address human-computer interaction and recommend a fanout (number of children for a given concept) no larger than ten and a taxonomy depth no higher than four levels. A larger fanout results in longer lists which have to be scanned to find the desired concept; a larger depth makes hierarchy traversal harder. Fanout and depth guidelines suggest taxonomies with a number of terminals ranging from 1,000 to 10,000, which are usually adequate for faceted taxonomies.

As an example of faceted design guidelines, consider a compound concept such as ‘15th century Italian paintings’. It can be split into its facets: a Location taxonomy (of which Italy is a descendant), a Time taxonomy (of which the fifteenth century is a descendant) and finally an Art taxonomy (of which painting is a descendant). The objects to be classified under the compound concept are classified under Location>Italy, Time>15th century and Art>Painting instead. The extensional inference rule establishes a relationship among these concepts and the compound concept can be recovered by zooming on any permutation of them.

In a conventional classification scheme, such as Dewey indexing [87], in which every object is classified under a single concept, a number of different concepts equal to the Cartesian product of the terminals in the three taxonomies must be defined.¹¹ Such a combinatorial growth either results in extremely large conceptual taxonomies or in a gross conceptual granularity [236]. In addition, faceted design coupled with dynamic taxonomies makes it simple to focus on a concept, e.g., 15th century, and immediately see all related concepts such as literature, painting, politics, etc., which are recovered through the extensional inference rule. In the compound concept approach, these correlations are unavailable because they are hidden inside the concept label.

An often overlooked consequence of faceted schemata is that the faceted design process breaks relationships into ‘primitive’ or ‘free-standing’ concepts which do not depend on other concepts [236]. In our experience, such primitive concepts are relatively stable in time, so that faceted schemata need little maintenance. It is the relationship among these primitive concepts that tends to vary over time, and dynamic taxonomies easily accommodate such variability by computing such relationships dynamically.

In summary, dynamic taxonomies require a simpler schema, dynamically adapt to new relationships, and simplify user discovery of unexpected relationships.

¹¹Usually, many of these concepts are meaningless (see Sect. 6.1).

1.4 Advantages

The advantages of dynamic taxonomies are especially significant in the areas of:

- user interaction, which is simple and natural;
- exploration, where dynamic and unexpected relationships are fully accounted for;
- schema design, where a faceted structure leads to minimal and flexible schemata, and
- search effectiveness, because dynamic taxonomies have an extremely fast convergence to small result sets.¹²

Dynamic taxonomies require a very light theoretical background: the concept of a taxonomic organization and the zoom operation, which seems to be very quickly understood by end-users. The user is effectively guided to reach his goal, because at each stage he has a complete list of all the concepts related to the current focus, which can be used to further refine his exploration. By construction, no empty results can occur because all the unrelated concepts which can cause empty results are automatically pruned and cannot be selected. Though guided, the user is in control of interaction and is free to explore the information base according to his interests: only the concepts which lead to dead ends are pruned, and inferred concept relationships are symmetric.

User control and the adaptive mechanism of dynamic taxonomies, which rapidly becomes clear and transparent, encourage the user to experiment and explore, and gives him the feeling that he has considered all the alternatives in reaching a result. This is confirmed by usability studies [134, 327] conducted on a corpus of art images. Despite slow response times, access through a dynamic taxonomy was shown to produce a faster overall interaction and a significantly better recall (both actual and perceived) than access through text retrieval.

We believe that significant advantages in ease of interaction are a result of using the same representation for querying (setting the focus) and summarizing, so that the user always deals with a single conceptual representation of the infobase. We call this property *self-adapting exploration*.¹³

The user-centered approach of dynamic taxonomies is inherently more acceptable to users than the system-centered approach of agent-mediated knowledge-based systems, so that dynamic taxonomies appear to be the model of choice for several important areas, such as medical diagnosis where knowledge-based systems have not been successful.

Exploration of complex infobases is free. Any combination of concepts (AND, OR, NOT) is supported by the corresponding set operations on their deep extensions, so that arbitrarily complex foci can be defined. Since the reduced taxonomy is dynamically produced, the model adapts to dynamic relationships among concepts (i.e., relationships which can vary in time) and lead to the discovery of unexpected

¹²See Sect. 3.2.

¹³Self-adapting exploration structures will be discussed in detail in Chap. 3.

relationships that even the designer might not be aware of. Most importantly, exploration is not restricted to conceptual manipulation only, but can be applied to any traditional search method (e.g. text retrieval, shape retrieval, etc.) because any subset of the universe can be summarized by dynamic taxonomies through a dynamically-created reduced taxonomy.

Since concepts can be dynamically combined at run-time, schemata for dynamic taxonomies are simple and minimal. They are based on subsumptions only and there is no need for compound concepts, which are the major cause of the combinatorial explosion of conventional taxonomies. No assumption is made on object contents, type or format, so that any type of heterogeneous information can be managed in a uniform way. Finally, in dynamic taxonomies, concepts are abstract entities, and no assumption is made on concept labels. Consequently, an appropriate architecture makes the support of multilingual access very easy because it only requires the translation of concept labels into a different language.

The advantages of dynamic taxonomies with respect to the convergence of exploratory patterns are dramatic. The analysis by Sacco¹⁴ [247] shows that 3 zoom operations on terminal concepts are sufficient to reduce a 10,000,000 object information base described by a compact taxonomy with 1,000 concepts to an average of 10 objects. Experimental data on a real newspaper corpus of over 110,000 articles, classified through a taxonomy of 1,100 concepts, reports an average 1,246 objects to be inspected by the user of a static taxonomy vs. an average 27 objects after a single zoom on a dynamic taxonomy.

Finally, the conceptual organization of dynamic taxonomies simplifies the gathering of user interests at a precise conceptual level by simply (and unobtrusively) monitoring the zoom operations issued and the concepts the user focuses on. At the same time, such an organization also allows the user to explicitly specify his own interests in terms of concepts so that precise push strategies can be implemented [231, 232, 242]. Recommendations can also be easily integrated, since reviews, popularity, etc. can be represented by specific facets in the dynamic taxonomy.

1.5 Application Areas

The main industrial application is currently e-commerce. Assisted product selection is a critical step in most large-scale e-commerce systems and the advantages in interaction are so significant as to justify the restructuring of most e-commerce portals, such as *Yahoo!* Shopping.

However, dynamic taxonomies have an extremely wide application range and a growing body of literature indicates that their adoption benefits most web applications. In addition to e-commerce, e-auctions, and e-catalogs, key areas such as e-government, human resources and job placement, news portals, multimedia,

¹⁴See Sect. 3.2.

medical guideline and diagnostic systems are being investigated and commercial solutions deployed.

Although most current applications fall in the object-seeking category, there is an emerging and growing interest in knowledge-seeking applications, while wisdom-seeking applications are currently at an initial stage. A detailed discussion of existing and emerging application areas is found in Chap. 9.

1.6 Faceted Search and Dynamic Taxonomies

Thus far, we have used the term *faceted search* as a synonym of *dynamic taxonomies*. However, faceted classification is just a design guideline, akin to normalization in relational databases, and the dynamic taxonomy model only requires a multidimensional taxonomy. Indeed, there are practical situations in which the violation of the orthogonal organization of facets is beneficial or required. For example, consider the topic taxonomy of a legal database organized for experts, and the same taxonomy organized for laymen. These two subtaxonomies are not facets as they are not orthogonal, but provide two different and useful access paths to the same infobase. So, it can be contended that the term faceted search is misleading since it focuses on an unessential feature. The concepts of extensional inference and of reduced taxonomies, which are a fundamental part of dynamic taxonomies, are not implied by (and do not require) a faceted classification.

In addition, faceted search is presented in literature through examples rather than being formally defined [248], and this has obviously caused a certain amount of confusion. Faceted search can be (and sometimes has been) taken at face value to mean any system based on a faceted classification. This covers very diverse solutions, ranging from the work of Prieto-Diaz [216], based on a faceted classification which can be composed by boolean operators but with no conceptual summarization capabilities, to early attempts by Amazon and Microsoft Knowledge Manager, where faceted subtaxonomies are completely independent and cannot be composed. Most importantly, it is impossible to reason in a rigorous way on features, extensions and challenges without a formal model.

Faceted search systems, though later, are subsumed by the more general dynamic taxonomy model. Although there is a clear evolution towards the more general model, some commonly found restrictions include, among others:

- attribute-value shallow taxonomies. In most systems, a two-level taxonomy is used, describing attributes with their values as immediate children. For example, attributes for a digital camera can be Price, Brand, Weight, etc. Each attribute A will have a number of children, each representing a value of A . This is a significant restriction with respect to the more general multi-taxonomy model supported by dynamic taxonomies, since it does not support any abstraction capabilities. However, this is a frequent approach because it allows a straightforward mapping

of database relations to a dynamic taxonomy in which the top-level facets are the attributes.¹⁵

- objects classified under terminal concepts only. This is really an implication of the previous restriction, because in attribute-value shallow taxonomies attributes are obviously abstract, and it makes no sense to classify an object directly under a specific feature. In general dynamic taxonomies this is not true. For example, a news item may be about an entire nation, a specific region, or a specific town.
- AND only refinement. Whereas dynamic taxonomies support all boolean operations on concepts, most faceted search systems only support refinement by a single selected concept. The lack of OR composition capabilities implies that the user is unable to focus on custom groups obtained by considering different concepts in the taxonomy as equivalent. As an example, consider a user interested in Balkan countries only: either a specific concept grouping such countries has been anticipated and exists in the schema or the user is unable to explore features for Balkan countries, unless OR composition is supported.

1.7 Book Roadmap

Dynamic taxonomies are user-centered and deal holistically with a number of different aspects which include modeling, user interaction, schema design, system implementation and performance. All of these aspects are fundamental in our approach, even though the final yardstick is whether the user can easily and effectively understand and use the model. Dynamic taxonomies are at the crossroads of several independent computer science research areas, and, for this reason, their initial acceptance by the research community was a relatively slow and difficult process because it was hard to classify them in the ‘appropriate’ conferences and journals, and contributions were usually and incorrectly considered from a single perspective only. On the contrary, the acceptance by the industrial community, once started, was very quick and pervasive.

For the purpose of exposition, the book has been divided into the following five main areas:

- *Modeling issues.* This is the focus of the next two chapters. Chapter 2 describes faceted taxonomy-based information sources. In Chap. 3, dynamic taxonomies and faceted search are compared to other techniques including information retrieval, OLAP, dynamic result clustering, static taxonomies, decision trees, formal concept analysis and description logics. Finally, Chap. 5 extends the model by addressing data mining, structured documents, and extended expressivity through logical information systems, and deals both with modeling and user interaction issues.
- *User Interaction issues.* Chapter 4 provides an extensive analysis of information presentation, interaction modes, user interface design patterns, and personalized faceted search.

¹⁵See Chap. 7.

- *Taxonomy design.* Chapters 6 and 7 focus on taxonomy design. Chapter 6 introduces the Compound Term Composition Algebra and addresses the integration of different and distributed taxonomy-based sources. Chapter 7 describes guidelines for schema design, and discusses the automatic generation of dynamic taxonomy schemata for structured databases, focusing on relational views and on E–R schemata. Finally, the automatic construction of dynamic taxonomy schemata from text infobases is discussed.
- *Architecture, implementation, and performance issues.* Effective user interaction with dynamic taxonomies requires real-time response for zoom operations, even for very large databases. Chapter 8 discusses architectural alternatives, and analyzes RDBMS-based and special implementations. It also discusses the composition of taxonomies with Logic components.
- *Applications.* Chapter 9 analyzes current and emerging application areas, including e-commerce, multimedia, e-government, human resource management, diagnostic systems, multidimensional file systems, and geographical information systems.

Chapter 2

Faceted Taxonomy-Based Sources

Yannis Tzitzikas

“μέγα βιβλίον μέγα κακόν”
(*Big book, big evil*)

Καλλίμαχος, 310 BC–240 BC

The objective of this chapter is to explain the underlying mathematical structure of faceted taxonomy-based sources and to provide some common notions and notations that are used in some parts of the book. Subsequently, and on the basis of the introduced formalism, this chapter describes the interaction between a user and an information source that supports dynamic taxonomies and faceted search. The comparison with other formalisms is described in Chap. 3 while a more detailed description of the interaction accompanied by examples of user interfaces is given in Chap. 4.

2.1 Introduction

Roughly, a *taxonomy* is a hierarchically-organized set of *terms*. A *term* can be a string but it can also be considered as a *concept*. A concept can be an abstract idea or a mental symbol, typically associated with a corresponding representation in a language or symbology. Therefore the same concept may have different representations in different natural languages (or even in the same language), as well as in different computer systems. For reasons of generality and simplicity, in this chapter we shall use *terms* to refer to both *concepts* and to plain names/values which are not necessarily concepts but are used for indexing/describing the objects of the domain of interest.

A *faceted taxonomy* is a set of taxonomies, each one describing the domain of interest from a different (preferably orthogonal) point of view [225]. Although exploratory search and dynamic taxonomies do not presuppose faceted taxonomies, i.e. they can be defined over single taxonomies, a faceted taxonomy has several advantages by comparison to a single taxonomy, such as conceptual clarity, compactness, and scalability (for more see [216, 300]).

Having a faceted taxonomy, each domain object (e.g., a book, a product, a Web page) can be indexed (or described) using a *compound term*, i.e., a set of terms

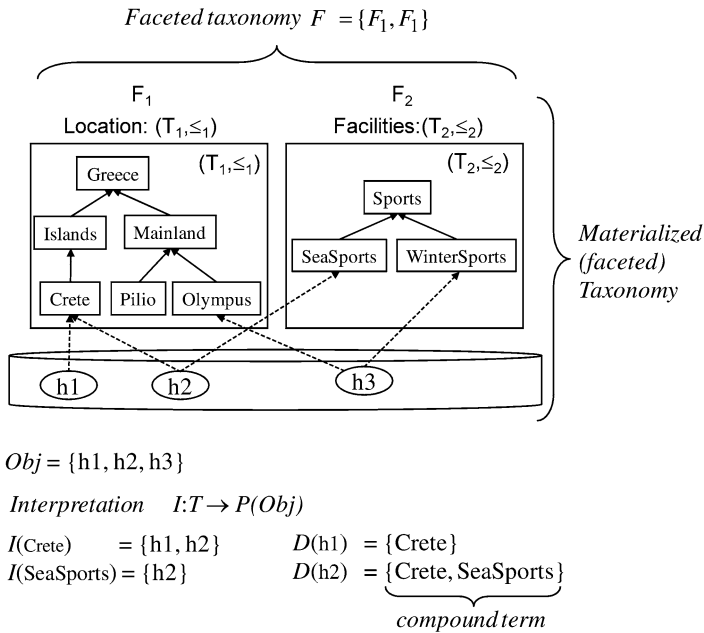


Fig. 2.1 A materialized faceted taxonomy

from one or several facets. A *materialized faceted taxonomy* (or *faceted taxonomy-based source*) is a faceted taxonomy accompanied by a set of object descriptions. Figure 2.1 shows a sample materialized faceted taxonomy that consists of two facets, namely Location and Facilities, and three indexed hotel Web pages.

2.2 Taxonomies and Partially-Ordered Sets

Definition 2.1 A *terminology* is a set of names, called *terms*.

Later on we will broaden the notion of term and we will show that in addition to names, we can consider numbers, sets, and intervals as terms.

Definition 2.2 A *taxonomy* is a pair (T, \leq) , where T is a *terminology* and \leq is a reflexive and transitive binary relation over T , called *subsumption*.

If a and b are terms of T and $a \leq b$ then we say that a is *subsumed* by b , or that b *subsumes* a . We also say that a is *narrower than* b , or that b is *broader than* a . For example, $Crete \leq Greece$ and $SeaSports \leq Sports$.

The basic principle (or guideline when designing taxonomies) is the following:

*the more narrow a term is,
the more information it carries
(i.e. the larger its intension is)
and the smaller its extension is.*

Roughly speaking, the extension of a term, refers to the set of domain objects upon which the term is applicable.

We say that two terms a and b are *equivalent*, and write $a \sim b$, if both $a \leq b$ and $b \leq a$ hold, e.g., Greece \sim Hellas.

Note that the subsumption relation is a *preorder* over T and that \sim is an equivalence relation over the terms of T . Moreover \leq is a *partial order* over the equivalence classes of terms. For reasons of simplicity we shall hereafter assume that \leq is a partial order (i.e. no cycles can be formed).

When using diagrams to depict a taxonomy, such as the ones of Fig. 2.1, term subsumption is indicated by a continuous-line arrow from the narrower term to the broader term. Note that we do not illustrate the entire subsumption relation but only a subset sufficient for generating it. In particular, we do not represent the reflexive, nor the transitive arrows of the subsumption relation (i.e. we illustrate the reflexive and transitive reduction of \leq). The resulting diagram is also called Hasse diagram if it has an upward orientation.¹

A linearly ordered set, e.g. the set $\{1, 2, 3, 4, 5\}$, can also be considered as a partially ordered set (i.e. the relation \leq over the set of integers is reflexive, transitive and antisymmetric), and thus it can also be considered as a taxonomy. Generalizing, any subset of a linearly ordered set (e.g. the set of real numbers) could be considered as a taxonomy. Furthermore, even a plain set (e.g. $\{a, b, c\}$) could (somehow abusively) be considered as a taxonomy (where \leq in such cases comprises only reflexive relationships). Also note that we can define intervals over linearly ordered sets, and intervals can be ordered according to the cover relation. It follows that a set of intervals ordered by the cover relation can also be considered as a taxonomy. Figure 2.2 shows some examples of taxonomies. However, the desired ordering depends on the semantics as it will be clarified later on. Taxonomies can be handcrafted or derived automatically from structured data (e.g. see Sects. 7.1 and 8.2), from textual sources (e.g. see Sects. 7.2 and 8.3.1), by composing other taxonomies and logics (e.g. see Sects. 6.1 and 8.5), etc.

2.3 Faceted Taxonomies

A faceted taxonomy comprises a set of facets, where each facet F_i has a name and as range a taxonomy. Table 2.1 shows all notations that are used in this chapter.

¹If $a < b$ then the point corresponding to a appears lower in the drawing than the point corresponding to b .

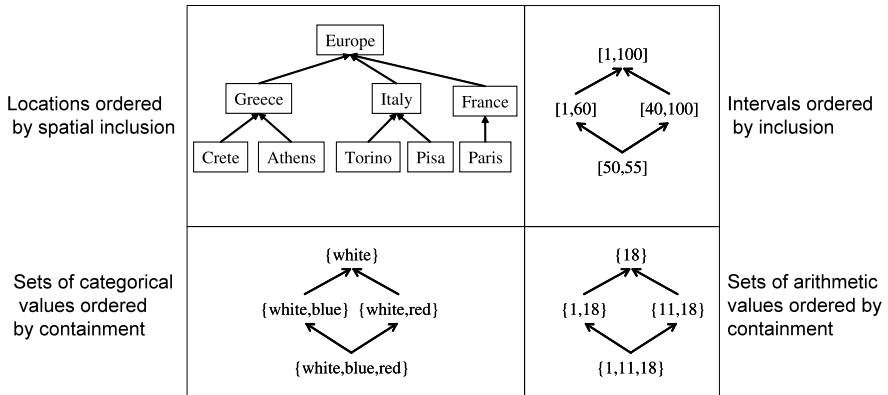


Fig. 2.2 Examples of taxonomies

Table 2.1 Basic notions and notations

Name	Notation	Definition
terminology	T	a set of names, called <i>terms</i>
subsumption	\leq	a preorder relation (reflexive and transitive)
taxonomy	(T, \leq)	T is a terminology, \leq a subsumption relation over T
broaders of t	$Br(t)$	$\{t' \mid t < t'\}$
narrowers of t	$Nr(t)$	$\{t' \mid t' < t\}$
direct broaders of t	$Br^{(1)}(t)$	$minimal_{\leq}(Br(t))$
direct narrowers of t	$Nr^{(1)}(t)$	$maximal_{\leq}(Nr(t))$
faceted taxonomy	$\mathcal{F} = \{F_1, \dots, F_k\}$	$F_i = \langle nm, (T_i, \leq_i) \rangle$, for $i = 1, \dots, k$ and all T_i are disjoint
compound term over T	s	any subset of T (i.e., any element of $\mathcal{P}(T)$)
object domain	Obj	any denumerable set of objects
interpretation of T	I	any function $I : T \rightarrow \mathcal{P}(Obj)$
model of (T, \leq)		
induced by I	\bar{I}	$\bar{I}(t) = \bigcup \{I(t') \mid t' \leq t\}$
materialized	(\mathcal{F}, I)	\mathcal{F} is a faceted taxonomy $\{F_1, \dots, F_k\}$, I is an interpretation of $T = \bigcup_{i=1, \dots, k} T_i$
extension of s		
in I and in \bar{I}	$I(s), \bar{I}(s)$	$I(s) = \bigcap \{I(t) \mid t \in s\}$ and $\bar{I}(s) = \bigcap \{\bar{I}(t) \mid t \in s\}$

Note that the same taxonomy can be used as range of more than one facets (for instance, for indexing flights we may have two facets named “from” and “to” whose range is the same taxonomy “Location”). However, by prefixing the name of each term with the facet name we can assume that the facet terminologies are all disjoint (as stated in Table 2.1).

A faceted taxonomy is actually a taxonomy. Suppose that $\mathcal{F} = \{F_1, \dots, F_k\}$ where $F_i = (T_i, \leq_i)$, for $i = 1, \dots, k$ and all T_i are disjoint. If we define:

$$T = T_1 \cup \dots \cup T_k$$

$$\leq = \leq_1 \cup \dots \cup \leq_k$$

it is evident that (T, \leq) is again a taxonomy. So the only difference is that the terminology and the subsumption relation of a faceted taxonomy is partitioned. The word “partitioned” raises questions of the form: *on what criteria such partitions should be based on, or what a facet is, or should be?* Unfortunately we cannot give a strict definition. In general, the decision of what a facet is or should be, is a modeling/methodological rather than technical issue, and a detailed discussion of such questions is beyond the scope of this chapter. Recall that facet-like structures occur in several disciplines: library and information science (originated from Ranganathan work [225] referring to *subdivisions* of a class hierarchy based on re-occurring features), object-oriented languages and systems (the notion of *class*), databases (the notion of *attribute*), formal context analysis (the notion of *scale*), software engineering (the notion of *aspect* in aspect-oriented analysis). Some of these are discussed in more detail in Chap. 3, while a discussion of this structuring mechanism in various domains is provided in [218]. Let us just include some comments coming from the area of library and information science. Vickery [313] compares the semantic model of human memory structures used by Lindsay and Norman [172] with the analysis of subjects by facet, used by himself and others in subject classification. For instance, they described “roles which characterize parts of an event” as:

*Action, Agent, Conditional, Instrument, Location,
Object, Purpose Quality, Recipient, Time.*

These correspond closely to some of the facets defined by Vickery [313] as being useful within a science and technology classification:

*Attributes, Object, Parts, Place, Processes, Properties, Substances, Time,
Recipient.*

Research based on facet analysis [95] has been able to define facets which may be labeled differently in different domains, but which are essentially transferable. The following examples are taken from knowledge bases built for research purposes using the hypertext system “NoteCards”. Notice that the labels in each row although different are semantically similar.

Generic labeling	Catering	Social skills
parts	ingredients	attitudes
processes	processes	processes
procedures	recipes	procedures
agents	equipment	people
properties	characteristics	situations

Facet analysis is also discussed in Sect. 7.1. Finally, we want to note that facet orthogonality is not a strict constraint.

2.4 Taxonomy-Based Sources

Let Obj be a finite set of *objects* called *domain*. We write “objects” (and not “items”) just to stress that each one of them has a unique identity. In our running example of Fig. 2.1, $Obj = \{h1, h2, h3\}$.

Definition 2.3 A *taxonomy-based source* S is a quadruple $S = \langle T, \leq, I, Q \rangle$ where:

- T is terminology
- \leq is a partial order over T called *subsumption*
- I is a function $I : T \rightarrow \mathcal{P}(Obj)$ called *interpretation* (where $\mathcal{P}(Obj)$ denotes the power set of Obj),
- Q is the set of all queries defined by the grammar $q ::= t \mid q \wedge q' \mid q \vee q' \mid \neg q \mid (q) \mid \epsilon$ where t is a term in T and ϵ the empty query.

Browsing and query answering in a taxonomy-based source S is founded on the notion of *model*.

Definition 2.4 An interpretation I is a *model* of a taxonomy (T, \leq) if for all t, t' in T , if $t \leq t'$ then $I(t) \subseteq I(t')$.

Definition 2.5 Given an interpretation I of T , the *model* of (T, \leq) *generated* by I , denoted \bar{I} , is given by:

$$\bar{I}(t) = \bigcup \{I(t') \mid t' \leq t\}$$

Notice that $I(t)$ is what is called the *shallow* extension of t , while $\bar{I}(t)$ is the *deep* extension of t .

Definition 2.6 Given two interpretations I, I' of T , we call I less than or equal to I' , and we write $I \sqsubseteq I'$, if $I(t) \subseteq I'(t)$ for each $t \in T$.

Clearly, \bar{I} is the minimal model of (T, \leq) that is greater than I with respect to \sqsubseteq .

Any interpretation I can be extended to an interpretation of queries as follows: $I(q \wedge q') = I(q) \cap I(q')$, $I(q \vee q') = I(q) \cup I(q')$, and $I(\neg q) = Obj - I(q)$, where $-$ denotes set difference.

Given a source $S = \langle T, \leq, I, Q \rangle$ and a query $q \in Q$, the *answer* of q is the set $\bar{I}(q)$.

Given an object $o \in Obj$, we can define its *description* according to an interpretation I , denoted by $D_I(o)$, as follows:

$$D_I(o) = \{t \in T \mid o \in I(t)\}$$

So $D_I(o)$ is a set of terms also called *compound term* (a compound term is any subset of T). In our example, $D_I(h1) = \{\text{Crete}\}$, while with respect to the model \bar{I} we have $D_{\bar{I}}(h1) = \{\text{Crete, Islands, Greece}\}$. We may refer to the latter as the *expanded* (or *complete*) *description* of $h1$. If $s = \{t_1, \dots, t_m\}$ is a compound term, then $I(s) = I(t_1) \cap \dots \cap I(t_m)$, i.e. we treat compound terms as conjunctions of terms.

Above we defined descriptions through interpretations. The reverse is also possible (i.e. to define interpretations through descriptions). Specifically, we may consider a description function $D : Obj \rightarrow \mathcal{P}(T)$ (where $\mathcal{P}(T)$ denotes the power set of T). From such a function D we can define an interpretation, denoted by I_D , as follows:

$$I_D(t) = \{o \in Obj \mid t \in D(o)\}$$

This approach is more close to the way data are usually entered by users in applications. For instance, when users annotate images with tags, they actually provide descriptions of these images which can then be used to define the interpretations of tags.

It is worth mentioning that the terminologies may not be defined explicitly a-priori, but they can consist of those terms (or values) that appear in the descriptions of the indexed objects. For instance, consider a facet with name `age` whose range is the set of all integers. Suppose we have three persons with ages 11, 18 and 81. In that case we can consider that the terminology of the facet `age` is the set $\{11, 18, 81\}$. In this example it would *not* be useful to assume any ordering for this set (e.g. $11 < 18 < 81$). In general this depends on the semantics (the more narrow a term is, the more information it carries so the smaller its extension is). For instance, consider a facet `priceRange` and two hotels whose price ranges are $[70, 80]$ and $[60, 100]$. In this case it is reasonable to assume that $[70, 80] < [60, 100]$ as the former is contained in the latter.

In brief the notion of taxonomy-based source as defined is very broad and it can be used to capture any attribute whose range is a partially ordered set. For instance, we can consider taxonomy-based sources as attribute-value pairs of:

- single quantitative attributes:
For example, `age = 18`.
- single categorical attributes:
For example, `color = red`.
- single hierarchically-ordered categorical attributes:
For example, `location = Athens` where `Athens` is a term of a taxonomy.
- multi-valued quantitative or categorical attributes:
For example, `age = {11, 18}` and `color = {white, blue}`. The semantics of the multiple values determine the ordering that should be assumed. For instance, if we assume that `color = {white, blue}` means that the color is both *white and blue* (e.g. the flag of Greece or Finland), then we should consider that:

$$\{\text{white, blue}\} \leq \{\text{white}\}$$

$$\{\text{white, blue}\} \leq \{\text{blue}\}$$

If we assume that $\text{age} = \{11, 18\}$ means that age is *either* 11 or 18, then we should consider that

$$\{11\} \leq \{11, 18\}$$

$$\{18\} \leq \{11, 18\}$$

- intervals:

For example, $\text{priceRange} = [80, 140]$. Again the semantics of intervals determine the ordering of the intervals, as it is described in the subsequent section.

2.5 On Intervals

An *interval*, specifically a closed interval of reals (of any linearly or partially ordered set in general), is any pair of elements $[a, b]$ such that $a \leq b$. We can define a partial order (\leq) over a set of intervals Φ as follows: $[a, b] \leq [c, d]$ iff $c \leq a$ and $b \leq d$. Clearly, this is the interval inclusion relation.

An interpretation of a set of intervals Φ over a set of objects Obj is again any function $I : \Phi \rightarrow \mathcal{P}(Obj)$. However we can distinguish two different meanings to an interpretation I , the *universal* and the *existential* one:

- (a) *Universal* meaning

Here, $o \in I(\phi)$ means that object o *appears* in the *whole* ϕ , or equivalently, that the entire interval ϕ applies to o . For instance, in the context of a variable *life* we can have $\text{Aristotle} \in I_{\text{life}}([-384, -322])$ as Aristotle lived from 384 B.C. until 322 B.C., or $\text{Einstein} \in I_{\text{life}}([1879, 1955])$. In the context of a variable *shoeSizes* we can have $\text{Timberland} \in I_{\text{shoeSizes}}([36, 44])$.

- (b) *Existential* meaning

Here, $o \in I(\phi)$ means that object o *appears somewhere* within ϕ . For instance, in the context of a variable *date* we can have $\text{Easter} \in I_{\text{date}}([\text{March}, \text{May}])$, or in the context of a variable *prices* we can have $\text{HolidayInn} \in I_{\text{prices}}([70, 200])$.

If I is an interpretation with universal meaning, it follows that if $o \in I(\phi)$ and $\phi' \leq \phi$ then o appears in the whole ϕ' too. This can be captured by the notion of universal model introduced next.

Definition 2.7 An interpretation of I of Φ is a *universal model* of (Φ, \leq) if for any $\phi, \phi' \in \Phi$, if $\phi \leq \phi'$ then $I(\phi) \supseteq I(\phi')$.

We can always extend an interpretation I of Φ to a universal model of (Φ, \leq) , denoted by \underline{I} , as follows:

$$\underline{I}(\phi) = \bigcup \{I(\phi') \mid \phi \leq \phi'\}$$

Clearly, \underline{I} is the minimal universal model of (Φ, \leq) that is greater than I with respect to \sqsubseteq .

If I is an interpretation with existential meaning, it follows that if $o \in I(\phi)$ and $\phi \leq \phi'$ then o also “appears” somewhere within ϕ' . This can be captured by the notion of existential model introduced next.

Definition 2.8 An interpretation of I of Φ is an *existential model* of (Φ, \leq) if for any $\phi, \phi' \in \Phi$, if $\phi \leq \phi'$ then $I(\phi) \subseteq I(\phi')$.

We can always extend an interpretation I of Φ to an existential model of (Φ, \leq) , denoted by \bar{I} , as follows:

$$\bar{I}(\phi) = \bigcup \{I(\phi') \mid \phi' \leq \phi\}$$

Clearly, \bar{I} is the minimal existential model of (Φ, \leq) that is greater than I with respect to \sqsubseteq .

We can call interval-based source, for short source, any triple (Φ, \leq, I) where Φ is a set of intervals, \leq is the inclusion relation over Φ and I is an interpretation of Φ .

We can query a source (Φ, \leq, I) in order to find objects that satisfy certain properties. Let us now introduce a basic query language for intervals. A query is any string derived by the following grammar, $q ::= U[a, b] \mid E[a, b]$, where $[a, b]$ is an interval. We will denote by Q the set of all queries. The queries of the form $U[a, b]$ are called *universal queries*, while the queries of the form $E[a, b]$ are called *existential queries*. Roughly, a universal query $U[a, b]$ seeks for objects that appear in the whole interval $[a, b]$, while an existential query $E[a, b]$ seeks for objects that appear somewhere in the interval $[a, b]$.

Let's now define query answering, i.e. define $ans(q)$ where $q \in Q$. Of course the answer of queries over a source (Φ, \leq, I) depends on the meaning of I . Below we discuss query answering for each possible case.

- universal queries over interpretations with universal meaning:

$$ans(U[a, b]) = \underline{I}([a, b]) = \bigcup \{I(\phi) \mid [a, b] \leq \phi\}$$

- existential queries over interpretations with existential meaning:

$$ans(E[a, b]) = \bar{I}([a, b]) = \bigcup \{I(\phi) \mid \phi \leq [a, b]\}$$

- universal queries over interpretations with existential meaning:

The answers of these queries are always empty. This is because from premises of the form $\exists X P(X)$ we cannot yield conclusions of the form $\forall X P(X)$, unless our intervals are points but this is a special case of limited practical interest.

- existential queries over interpretations with universal meaning:

Let us first introduce some auxiliary definitions. Two intervals $[a, b]$ and $[c, d]$ are *overlapping*, if $c \leq b$. The intersection of two intervals $[a, b]$ and $[c, d]$, denoted by $[a, b] \cap [c, d]$, is defined as: $[a, b] \cap [c, d] = [c, b]$ if they are overlapping, and $[\]$ otherwise. One can easily see that if q is an existential query, then the answer of q (over a universal interpretation) is given by:

$$ans(E[a, b]) = \bigcup \{I(\phi) \mid \phi \text{ overlaps } [a, b]\}$$

2.6 Modeling Interaction

As interaction is of prominent importance, this section introduces the basic notions for describing interaction. It describes the essentials from various works and systems either called dynamic taxonomies or not (e.g. [213, 300, 302]). However a more detailed treatment of the subject is available in Chaps. 3 and 4.

In brief, the user explores or navigates the information space by setting and changing his/her *focus*. The notion of focus can be intensional or extensional. Specifically, any set of terms, i.e. any conjunction of terms (else called compound term) is a possible *focus* (more in Chap. 4). Of course we could generalize and consider any query as a possible focus. However hereafter we shall concentrate on foci that are conjunctions of terms. For example, the initial focus could be the empty compound term, or the top term of a facet. However, and as we shall see in Sect. 2.7 (as well as in the chapter that describes systems, e.g. in Sect. 8.3.1), the user could also start from an arbitrary set of objects. In that case we could say that the focus is defined extensionally.

For reasons of minimality, we shall hereafter consider foci that are *redundancy free*. A focus ctx (i.e. $ctx \subseteq T$) is redundancy free if and only if $ctx = \text{minimal}_{\leq}(ctx)$. For example, $ctx = \{Greece, Crete\}$ is not redundancy free because $\text{minimal}_{\leq}(ctx) = \{Crete\}$.

The *contents (or extension) of a focus ctx* , is the set of objects $\bar{I}(ctx)$. We can refine this notion and distinguish the *shallow contents* $I(ctx)$, from the *deep contents* $\bar{I}(ctx)$.

Consider the materialized faceted taxonomy shown in Fig. 2.3 (notice that the third facet comprises intervals that have an existential meaning). In this case we

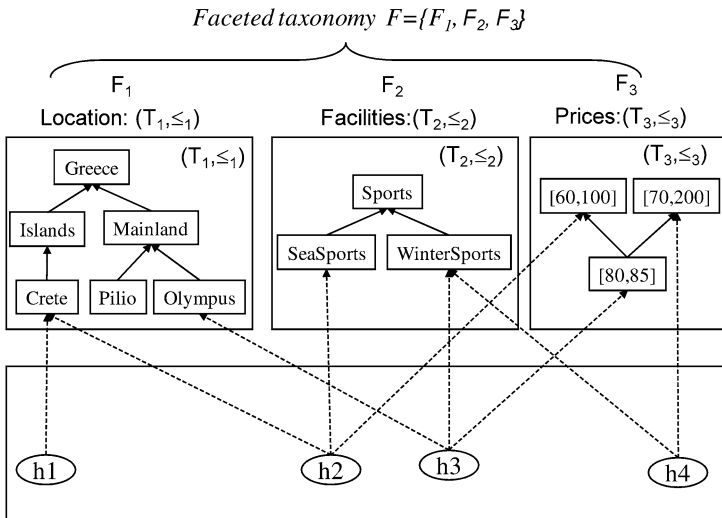


Fig. 2.3 A materialized faceted taxonomy comprising three facets

have:

$$I(\{\text{Islands}, \text{SeaSports}\}) = \emptyset$$

$$\bar{I}(\{\text{Islands}, \text{SeaSports}\}) = \{\text{h2}\}$$

2.6.1 Zoom Points

Given a context ctx , we can define the set of all zoom points with respect to a particular facet F_i , denoted by $AZ_i(ctx)$ as follows:

$$AZ_i(ctx) = \{t \in T_i \mid \bar{I}(ctx) \cap \bar{I}(t) \neq \emptyset\}$$

These zoom points can be categorized to various categories, e.g. zoom-in/out/side points, as shown in Table 2.2 and are discussed in the subsequent sections.

When building GUIs, an area is usually dedicated to each facet and the zoom points with respect to a facet F_i are actually those terms of T_i that should be shown in that area (these are the extensionally related terms as introduced in Chap. 1). Each

Table 2.2 Interaction notions and notations

Name	Notation	Definition
focus	ctx	any subset of T such that $ctx = \text{minimal}_{\leq}(ctx)$
focus projection on a facet i	ctx_i	$ctx_i = ctx \cap T_i$
Kinds of zoom points w.r.t. a facet i while being at ctx	Notation	Definition(s)
zoom points	$AZ_i(ctx)$	$= \{t \in T_i \mid \bar{I}(ctx) \cap \bar{I}(t) \neq \emptyset\}$
zoom-in points	$Z_i^+(ctx)$	$= AZ_i(ctx) \cap Nr(ctx_i)$
immediate zoom-in points	$Z_i(ctx)$	$= \text{maximal}_{\leq}(Z_i^+(ctx))$ $= AZ_i(ctx) \cap Nr^{(1)}(ctx_i)$
zoom-side points	$ZR_i^+(ctx)$	$= AZ_i(ctx) \setminus \{ctx_i \cup Nr(ctx_i) \cup Br(ctx_i)\}$
immediate zoom-side points	$ZR_i(ctx)$	$= \text{maximal}_{\leq}(ZR_i^+(ctx))$
zoom-out points	$ZO_i^+(ctx)$	$= Br(ctx_i)$
immediate zoom-out points	$ZO_i(ctx)$	$= Br^{(1)}(ctx_i)$
Restriction over an object set	Notation	Definition(s)
restricted object set	A	any subset of Obj
reduced interpretation	I'	$I'(t) = I(t) \cap A$
reduced terminology	T'	$= \{t \in T \mid \bar{I}'(t) \neq \emptyset\}$ $= \{t \in T \mid \bar{I}(t) \cap A \neq \emptyset\}$ $= \bigcup_{o \in A} Br(D_I(o))$

zoom point t_x is usually accompanied by a number (related count) that indicates the number of objects that will be obtained if the user selects that point. Specifically that number equals the cardinality of the set $\bar{I}(ctx) \cap \bar{I}(t_x) = \bar{I}(ctx \cup \{t_x\})$, which is certainly greater than zero.

The zoom points can be ranked according to various criteria like, number of indexed objects, user preferences, popularity, usage workload. In addition, other criteria can be employed to suppress the visibility of some points. For instance, we may hide those zoom points leading to contexts with contents size below a predefined threshold, or we may decide to present only the top- K zoom points for each facet. More details are given in Chap. 4.

2.6.2 Zoom-in

Now we introduce elements allowing the refinement of a focus. To this end we introduce the notion of *zoom-in points*. A zoom-in point is actually a term that indicates where the user could zoom in.

Given a focus ctx , we can define its *projection* to a facet F_i , denoted ctx_i , as follows $ctx_i = ctx \cap T_i$.

For example, assume we use the number 1 for Location, the number 2 for Facilities, and 3 for Prices. If $ctx = \{\text{Islands}, \text{SeaSports}\}$ then

$$\begin{aligned} ctx_1 &= \{\text{Islands}\}, \\ ctx_2 &= \{\text{SeaSports}\} \text{ and} \\ ctx_3 &= \emptyset. \end{aligned}$$

Now we can define the zoom-in points with respect to a particular facet F_i . Specifically we will define the *immediately narrower* zoom-in points (of course these points can have more narrower zoom-in points, and so on). Consider a focus ctx and suppose that $ctx_i \neq \emptyset$. The *candidate zoom-in points* with respect to F_i , denoted by $CZ_i(ctx)$, are defined as:

$$CZ_i(ctx) = Nr^{(1)}(ctx_i)$$

i.e. the direct narrower terms of ctx_i . In our example we have:

$$\begin{aligned} CZ_1(\{\text{Mainland}, \text{WinterSports}\}) &= \{\text{Pilio}, \text{Olympus}\} \\ CZ_2(\{\text{Mainland}, \text{WinterSports}\}) &= \emptyset \\ CZ_3(\{\text{Mainland}, \text{WinterSports}\}) &= \emptyset \end{aligned}$$

The above definition can also be applied in cases where $|ctx_i| > 1$, by assuming that $Nr^{(1)}$ is defined also for sets of terms.²

²If $S \subseteq T$ then $Nr^{(1)}(S) = \bigcup_{t \in S} Nr^{(1)}(t)$.

We can now filter out those points that would yield empty contents. So we can define the (immediately narrower) *zoom-in points* as:

$$Z_i(ctx) = \{t_x \in CZ_i(ctx) \mid \bar{I}(ctx) \cap \bar{I}(t_x) \neq \emptyset\}$$

In our example we have:

$$CZ_1(\{\text{Mainland}, \text{WinterSports}\}) = \{\text{Pilio}, \text{Olympus}\}$$

$$Z_1(\{\text{Mainland}, \text{WinterSports}\}) = \{\text{Olympus}\}$$

So $Z_i(ctx)$ comprises those terms of T_i that should be shown in the UI area dedicated to facet F_i when user focus is ctx .

When the user selects a zoom-in point t , then the current focus is updated, i.e. $ctx' = ctx \cup \{t\}$. If we want ctx' to be redundancy free we should remove those elements of ctx that are broader than t . Subsequently, all new zoom-in points are computed and presented, and so on.

Of course we could pre-compute (and even show in the UI area in a hierarchical way) all zoom-in points (not only the immediately narrower ones). Specifically, the set of all zoom-in points with respect to a facet F_i are defined as:

$$Z_i^+(ctx) = AZ_i(ctx) \cap Nr(ctx_i)$$

and of course $Z_i(ctx)$ comprises the maximal elements of $Z_i^+(ctx)$:

$$\begin{aligned} Z_i(ctx) &= \text{maximal}_{\leq}(Z_i^+(ctx)) \\ &= AZ_i(ctx) \cap Nr^{(1)}(ctx_i) \end{aligned}$$

2.6.3 Zoom-Side

Now we introduce another kind of zoom points. This kind of points is useful for taxonomy-based sources that satisfy at least one of the following conditions:

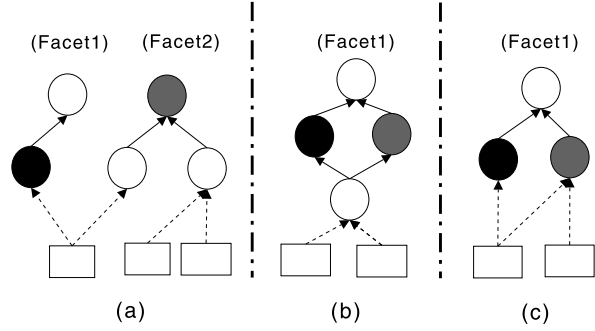
- (a) comprise more than one taxonomy (i.e. they are faceted taxonomies),
- (b) comprise a taxonomy that is not a tree, e.g. it is a DAG (Directed Acyclic Graph),
- (c) multiple classification (i.e. an object can be indexed with more than one terms) is allowed with respect to at least one facet.

If none of the above conditions hold, then this notion is not useful. Figure 2.4 highlights the corresponding conditions. In each diagram, the focus term is black-colored, while the (maximal) zoom-side terms are grey-colored.

Definition 2.9 From a materialized faceted taxonomy $M = (\mathcal{F}, I)$ we can define a symmetric binary relation \rightleftharpoons over T , called *extensionally related*, as follows

$$t \rightleftharpoons t' \quad \text{iff} \quad \bar{I}(t) \cap \bar{I}(t') \neq \emptyset \text{ and } t \parallel t'$$

Fig. 2.4 Cases where zoom-side points are useful



where $t \parallel t'$ means that t and t' are *incomparable* with respect to \leq (i.e. neither $t \leq t'$ nor $t' \leq t$).

Notice that this relation is symmetric, but not transitive. Let $R_i(t)$ denote the terms of T_i that are extensionally related to t , i.e.

$$R_i(t) = \{t' \in T_i \mid t \rightleftharpoons t'\}$$

We can now define the *zoom-side points* of a context ctx w.r.t. a facet F_i , denoted by $ZR_i(ctx)$, as follows:

$$ZR_i(ctx) = \text{maximal}_{\leq}(\{t \in T_i \mid \bar{I}(t) \cap \bar{I}(ctx) \neq \emptyset \text{ and } t \parallel ctx_i\})$$

Notice that if ctx is a single term, say $ctx = t$, then $ZR_i(t) = \text{maximal}_{\leq}(R_i(t))$.

Above we used the maximal elements just to reduce the number of points (and thus avoid cluttering the user screen). Of course we could define the set of all zoom-side points as follows:

$$\begin{aligned} ZR_i^+(ctx) &= AZ_i(ctx) \setminus \{ctx_i \cup Nr(ctx_i) \cup Br(ctx_i)\} \\ ZR_i(ctx) &= \text{maximal}_{\leq}(ZR_i^+(ctx)) \end{aligned}$$

Some examples over our running example follow.

$$\begin{aligned} ZR_1(\{\text{Mainland}\}) &= \emptyset \\ ZR_2(\{\text{Mainland}\}) &= \{\text{Sports}\} \\ ZR_3(\{\text{Mainland}\}) &= \{[60, 100], [70, 200]\} \end{aligned}$$

2.6.4 Zoom-out

The user can also *zoom out* by deselecting one term t of the focus. In that case t is replaced by its direct broader term(s) i.e. by $Br^{(1)}(t)$. Alternatively, the user may

select any broader term t' of t (in that case t is replaced by t'), or even remove t without replacing it with any other term. We could define:

$$\begin{aligned} ZO_i^+(ctx) &= Br(ctx_i) \\ ZO_i(ctx) &= Br^{(1)}(ctx_i) \end{aligned}$$

To capture the cases where the user wants to delete all terms coming from a particular facet, we could assume that each facet has a top element denoted by \top that subsumes all other terms (so \top is the maximum element of $Br(ctx_i)$ for every ctx_i).

2.7 Restriction

As faceted exploration can be combined easily with other access methods (e.g. information retrieval queries, structured queries, or application-specific queries), the user could start interacting not only by selecting some terms (i.e. by specifying a focus), but through a set of objects, e.g. the objects returned by a full text query. To this end in this section we introduce a notion useful for capturing such scenarios.

Let $M = (\mathcal{F}, I)$ be a taxonomy-based source. Let A be a subset of Obj ($A \subseteq Obj$) which could be the result of an arbitrary access method. Below we will define the *restriction of M on A* , hereafter denoted by $(\mathcal{F}, I)|_A$.

The *restriction of M on A* , i.e. $(\mathcal{F}, I)|_A$, is again a materialized faceted taxonomy, and let us write $(\mathcal{F}, I)|_A = (\mathcal{F}', I')$. It comprises a restriction of the interpretation function I and a restriction of the faceted taxonomy \mathcal{F} . The later is the *reduced taxonomy* (as mentioned in Chap. 1).

Specifically, the interpretation I' is an interpretation that is smaller than I , i.e. $I' \sqsubseteq I$ (according to Def. 2.6). In particular, I' is defined as follows:

$$\forall t \in T, \quad I'(t) = I(t) \cap A$$

So the range of I' is the power set of A (and not the power set of Obj as it is for I).

As mentioned in Sect. 2.4, from a given interpretation I , we can define a description function, denoted by $D_I(o)$ as follows:

$$\forall o \in Obj, \quad D_I(o) = \{t \in T \mid o \in I(t)\}$$

and vice versa (i.e. from a description function D we can define an interpretation I), as in the example of Table 2.3.

Table 2.3 From Interpretations to Descriptions and vice versa

I	D_I
$I(t1) = \{obj1\}$	$D(obj1) = \{t1, t2\}$
$I(t2) = \{obj1, obj2\}$	$D(obj2) = \{t2\}$

The domain of the function D_I is the set Obj . We can restrict the domain of D_I on A , i.e. we can define the function $D_{I|A}$ (where $D_{I|A}$ denotes the restriction of the domain of D_I on A). It is equivalent to say that the interpretation I' of the restriction of M on A , is the interpretation obtained by the description function $D_{I|A}$.

Now the *reduced taxonomy* \mathcal{F}' comprises a terminology T' ($T' \subseteq T$) defined as follows

$$T' = \{t \in T \mid \bar{I}(t) \cap A \neq \emptyset\}$$

Equivalently,

$$T' = \bigcup_{o \in A} Br(D_I(o))$$

i.e. it comprises those terms that are associated with the objects in A plus all broader terms of these terms. We could denote this terminology by $T_{|I,A}$.

Definition 2.10 The *restriction of a materialized faceted taxonomy* $M = (\mathcal{F}, I)$ over a set of objects A , denoted by $(\mathcal{F}, I)_{|A}$, is again a materialized faceted taxonomy, comprising a reduced taxonomy with terminology $T' = \{t \in T \mid \bar{I}(t) \cap A \neq \emptyset\}$ and an interpretation I' such that $I'(t) = I(t) \cap A$ for each $t \in T$.

For example, consider the materialized faceted taxonomy shown in Fig. 2.1. If $A = \{h2\}$ then $I'(\{\text{Crete}\}) = \{h2\}$ and $\bar{I}'(\{\text{Islands}\}) = \{h2\}$. Regarding the reduced terminologies of T_1 and T_2 , we have:

$$T'_1 = T_1 - \{\text{Mainland, Pilio, Olympus}\}$$

$$T'_2 = T_2 - \{\text{WinterSports}\}$$

where “ $-$ ” denotes set difference.

Chapter 3

Comparison with Other Techniques

Giovanni Maria Sacco, Sébastien Ferré,
and Yannis Tzitzikas

“Every day I remind myself that my inner and outer life are based on the labors of other men, living and dead, and that I must exert myself in order to give in the same measure as I have received and am still receiving.”

Albert Einstein, 1879–1955

We compare dynamic taxonomies with the other main approaches to information access and discuss analogies and differences. The approaches we analyze range from traditional retrieval paradigms, such as queries on structured data, to the most recent approaches, including the current effort on the Semantic Web:¹

- queries on structured data, and OLAP data analysis techniques;
- information retrieval on textual material, and recent extensions such as dynamic clustering and term suggestion;
- static taxonomies, and multidimensional taxonomies without concept composition;
- decision trees;
- Formal Concept Analysis (FCA), and
- Description Logics and the Semantic Web.

3.1 Structured Access and Information Retrieval

Traditional techniques such as queries on structured databases or information retrieval queries on textual unstructured data address the retrieval of information on the basis of precise specifications, and are consequently used for focused retrieval rather than for exploration. However, extensions giving limited exploration capabilities have been recently proposed and a comparison with dynamic taxonomies is relevant.

For comparison purposes, we introduce two models of interaction: the retrieval model and the exploration model. The *retrieval model*, has two components:

¹Dynamic taxonomies were influenced by the minimalistic philosophy of the Fact Model, an early semantic data model [235].

- the query component, which is used to formulate a query, i.e., a specification which must be satisfied by the objects retrieved, and
- the result component, which is used to present the result to the user.

The user submits a query through the query component, and the state of the system changes by computing a new result component for the query. The result component is empty if no object satisfies the query.

The query component can be free-form, structured, or guided. In a free-form query component no system assistance is given to the user. The best known example of free-form queries is the Google query page [148]: a text field where the user can input any search expression with no system assistance. For structured data, free-form queries allow the user to directly enter queries in a specific query language such as SQL [69].

Free-form queries are quite unsatisfactory from the interaction perspective, especially when accessing structured data. In such cases, a free-form query interface requires an intimate knowledge of the underlying conceptual schema, which cannot be expected from a casual user. For this reason, applications which access structured data normally use a structured-form query component, in which different query fields are presented to the user as an input form. The query component accepts the values entered by the user and generates the actual query. This approach is often used for text-retrieval applications as well (e.g., Google’s Scholar [122]). The motivation here is not to simplify user interactions, but rather to make users queries more precise. As shown in Fig. 3.1, entering a search term in the Author field restricts the query to a specific field within each document rather than to the entire text.

In guided query components, users do not enter query terms or expression directly, but select them from a predefined set of choices. Each choice available to the user is called a *selector* in the following. The best known example of a guided query component is *Yahoo!’s* [321], where the query component is organized as a taxonomy of topics, with each topic being a selector. A simpler approach which

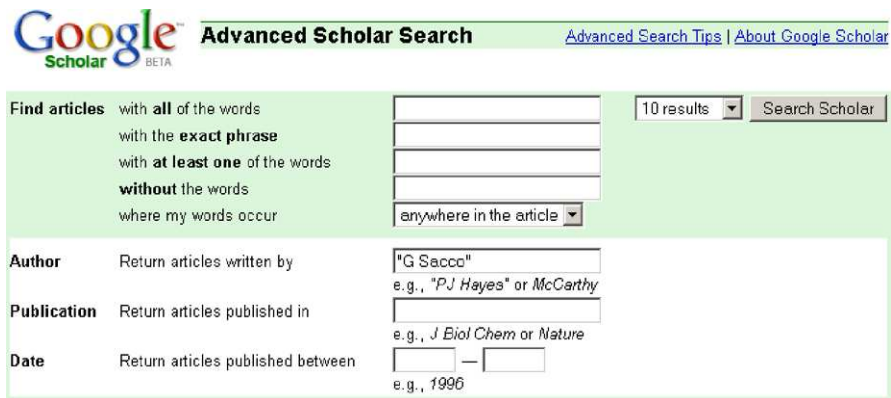


Fig. 3.1 Google Scholar: different query fields for Author, Publication, Date

still qualifies as a guided query component is a query form in which users can only select items from widgets such as combo boxes, radio, or checkbox groups and cannot enter values directly. Since human-computer interaction research indicates that *recognizing and selecting* is considerably easier than *remembering and entering*, guided query components are desirable where appropriate.

The result component presents the retrieved objects, by listing them by their ‘title’ and usually supports ordering operations on specific fields and, in some cases, on multiple fields. In text-retrieval or information retrieval applications, objects in the result can also be ordered by decreasing relevance with respect to the query.

The only component in the retrieval model modified by a user query is the result component.

In the second model we introduce, the *exploration model*, a query submitted by the user modifies, in general, the query component itself. The query component is represented by *self-adapting exploration structures* (SAES) which present a uniform view of the infobase consisting of multiple choices (*selectors* or *filters*) and use the same structure to set interest sets (*foci*) and summarize results. SAES’s have two fundamental properties:

- selectors are used to set interest sets (foci), i.e., to ‘query’ as in the guided retrieval model;
- selectors are used to summarize results. This is a fundamental point and we claim that there are no effective exploration capabilities in systems which do not support summaries of the result set.

In addition we characterize these structures according to six dimensions

- *completeness of iteration*. Iteration is complete if the process of selecting a focus and summarizing it can be iterated at will, or at least until a singleton result set is obtained. Incomplete iteration results in the impossibility of further restricting the infobase;
- *completeness of reach*. Reach is complete if it can be guaranteed that there are no objects in the infobase which are not reachable through at least one selector. Incomplete reach cause a loss of potentially relevant documents because some objects are never considered during exploration. Consequently, reach-incomplete structures should never be used as the only access path to an infobase;
- *self-sufficiency*. A SAES is self-sufficient if the use of selectors is always supported, and it is the only required operation although other types of queries (e.g., text retrieval) can be supported. Non self-sufficient structures often use faster query methods in the initial state (i.e., the entire infobase) as the only querying method. This is done primarily for performance reasons, and it results in asymmetric interactions;
- *no-zero-result*. It is a property satisfied when selectors which do not describe the result are pruned from summaries, or made non selectable. Consequently, the use of a selector is guaranteed never to produce an empty result;
- *expressivity*. In the simplest case, only one selector can be chosen. In the most general case, any boolean combination of selectors can be used;

- taxonomic vs. flat organization. In the first case, selectors can be organized according to a taxonomy so that the user can select the appropriate level of abstraction and user orientation among selectors is simpler.²

It is straightforward to show that dynamic taxonomies are SAES and that, additionally, they are taxonomically organized, are self-sufficient, no-zero result and complete from the perspective of iteration and reach.

Most SAES can be mapped to dynamic taxonomies. For example, consider the familiar tag cloud access structure [127], where the idea is to use the most frequent tags occurring in document descriptions as selectors. Normally more frequent tags are larger, but this is a human-computer interaction issue. Current systems do not provide any summary capabilities and, therefore, they are not self-adapting exploration structures.

We can extend the access structure by summarizing the result of a tag query through a tag cloud built for the documents in the result only. At this point, we have a SAES: the selectors are the tags in the cloud which are used both to ‘query’ and to summarize. Although this structure is complete from an iteration perspective, it is incomplete from perspective of reach. In general, only the k most frequent tags are used and, therefore, there is no guarantee that all the objects in the infobase are reachable through a query on the structure. Finally, tag clouds normally use a flat organization rather than a taxonomic one.³

However, there are SAES which cannot be mapped to DTs. Geographic maps,⁴ for instance, are SAES because they can be used (a) to select objects by using the geographic coordinates of the displayed portion of the map as a selector or filter, and (b) to summarize retrieved objects, by showing them on the displayed portion of the map itself. However, SAES geographic maps cannot be mapped to dynamic taxonomies.

3.1.1 *Queries on Structured Data*

As mentioned above, access to structured data is traditionally implemented using the retrieval, rather than the exploration, model.

Due to human factors, the interaction is usually performed through predefined query forms, rather than the query language itself. Widgets, such as radio and check buttons, are used to show admissible and selectable values, and implement selectors.

²SAESs supporting taxonomic organizations subsume flat SAESs because a flat organization can be represented by a trivial taxonomy comprising a root with all the flat selectors as immediate children.

³An extended tag cloud can be mapped to a dynamic taxonomy by considering tags as concepts in a shallow taxonomy. As we will show in Sect. 5.7.2, SAESs can be used as selectors in dynamic taxonomies. Specifically, the integration of tag clouds with DTs will be discussed.

⁴See also Sect. 5.7.

Some primitive summary information can be provided by sorting the result according to the values of one or more attributes. Summarizing by sorting is obviously much less effective than summarizing through a taxonomy, and fails to give a precise view of the result except for small result sets for which summaries are of little assistance. Most importantly, this type of access does not satisfy our definition of a SAES. Specifically, selectors are not used for summaries, and users refine their queries by adding new selectors in an unconstrained way. This can obviously lead to empty result sets.

3.1.1.1 OLAP and Data Warehousing

Introduced in the early 90's [70], OLAP (on line analytical processing) is used for the analysis of transaction data. The initial motivation for OLAP was the need for reports that were more complete, more sophisticated and easier to construct than those offered by existing systems. Central requirements for these flexible reporting capabilities are the extraction of data from large databases and aggregation capabilities according to different perspectives. These initial motivations have gradually expanded to encompass a wide area called business intelligence that extends the analysis capabilities of OLAP towards data mining and what-if analyses. Since data analysis usually requires historical data and massive computations, a clear separation between the structures needed for daily operations (OLTP, online transaction processing) and the structures needed for OLAP analysis (data warehouses) is required.

Data are represented through multidimensional tables, called *fact tables*. Each dimension of the table represents a different perspective that can be useful for analysis, e.g., product, location, date. Each dimension in the table can be hierarchically organized so that data aggregation can be performed. For example the Location dimension can be organized by states/provinces, nations and continents. Each fact in a table records an event, such as a sale, that contains appropriate values for all the dimensions and, in addition, values for additional attributes that are useful to describe the fact, e.g., price, quantity, etc. These additional attributes are called variables, properties, metrics, or indicators.

In a relational representation, the dimensions are the primary key of the fact relation: Fact(Product, Location, Date, Price, Quantity). In a geometric interpretation, a specific combination of values for dimensions identifies a cell. The entire n -dimensional hypercube has a number of cells equal to the Cartesian product of all the dimensions, so that, in practice, the vast majority of cells is empty. In both interpretations, the fact is the atomic level of aggregation for events. In the sale example with dimensions Product, Location and Date, the fact aggregates all the sales for a specific product at a specific location on a specific date: all the sales for product P at location L on date D will be aggregated into a single fact, and the individual sale transactions will not be available for analysis.

The central operations in OLAP are restriction, which is used to select a subspace for subsequent analysis, and aggregation, which is used to compute aggregates for

the metrics according to the dimensions selected for analysis. As mentioned before, the initial motivation and still the main application of OLAP is the flexible production of reports, and the import of data into spreadsheets in order to perform additional analysis.

In order to restrict a fact table, OLAP offers a number of primitive operations:

- *Roll-up* and *drill-down*: increases (roll-up) or decreases (drill-down) the level of aggregation for a dimension. For example, assume that the current level of aggregation for Location is Nation; this means that sales aggregated by Nations are currently shown. A roll-up will increase the current level to Continents, while a drill-down will decrease it to states/provinces;
- *Slice and dice*: allows to fix a value for one or more dimensions, thereby reducing both the dimensionality of the table, and the number of facts to be considered. For instance, we could set a specific location X, which transforms our three-dimensional sale table into a two-dimensional table on product and date;
- *Filtering*: reduces the number of facts to be considered through a selection criterion;
- *Pivoting* or *Rotation*: changes the dimensional orientation of data. In our example, data is initially aggregated by Product, Location, Date. By rotating, we can aggregate, for instance, by Location, Date and Product.

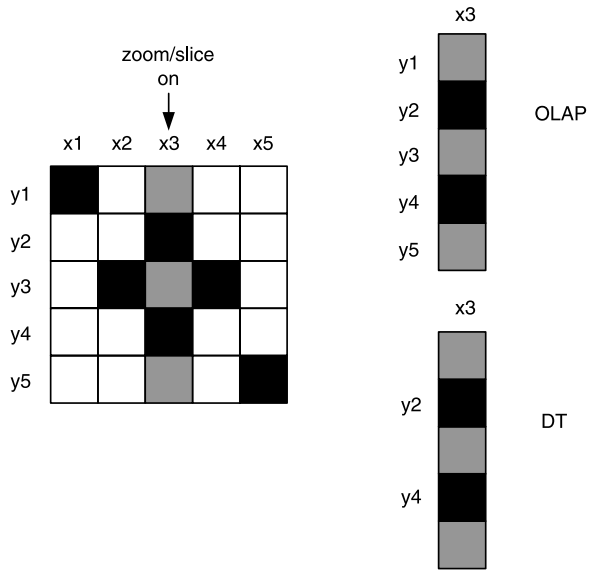
Although OLAP produces flexible and sophisticated summaries of results, it uses the retrieval, rather than the exploration, model. Consequently, OLAP *per se* does not support exploration.⁵ However, the underlying model is similar [247]. In fact, OLAP is based on a multidimensional model and explicitly exploits multidimensionality to support extracting and viewing data from different perspectives. A dimension in OLAP is equivalent to a facet in the dynamic taxonomy model. Both models allow hierarchical dimensions/facets. Such a substantial similarity suggests that design strategies for OLAP data warehouses, such as star schemata [62], can be used in dynamic taxonomies as well.

Figure 3.2 shows the difference between OLAP and DTs with respect to exploration. We have a two-dimensional hypercube, or equivalently, a two facet shallow taxonomy, where black cells are non-empty. A slice operation (OLAP) on a value x_3 for dimension x and a zoom operation (DT) on a value x_3 for facet x retrieve the same objects. However, the slice operation has no effect on the other dimension y , whereas the zoom operation prunes from facet y all the concepts which have an empty intersection with the cells with value x_3 .

It should be noted that also a relation in the relational model is inherently multidimensional. Each attribute defines a different dimension, so that each record can be seen as a point (or a hypercube cell) in the n -dimensional space defined by the n attributes of a relation. This interpretation will be used in Chap. 7 to derive translations from relational views to dynamic taxonomies. As mentioned before, fact tables can be represented by relations in which the dimensions of the fact table are

⁵An integration of DTs exploration capabilities with OLAP summary capabilities is proposed in [37].

Fig. 3.2 A zoom/slice operation on a two-facet taxonomy/two-dimensional hypercube, with the result for OLAP and DT. Black cells are non-empty



the primary key. So OLAP only allows the attributes in the primary key to be used as dimensions, even though any attribute could be used.

3.1.2 Information Retrieval

Information retrieval systems allow the retrieval of objects on the basis of their contents. Although the focus is on textual documents, information retrieval techniques can be applied to any type of objects described by metadata or tags or textual descriptions. Two major models were proposed in the past: the boolean model and vector space model. The first model is an exact retrieval model which retrieves sets of objects, whereas the second model is a best match retrieval model which retrieves ranked lists of objects.

In the basic boolean model, query terms are composed in a boolean expression and the system retrieves all the documents satisfying that expression. We are not concerned here with the system architecture and the performance of systems based on this model [28], but we remark that the result list is a flat list, because of boolean evaluation: each document either satisfies the expression and it is retrieved, or it does not and it is not.

The vector space model, proposed by Salton [258], characterizes both documents and queries as vectors in an n -dimensional space. Each dimension in this space corresponds to a term found in the corpus,⁶ so that the space considered has usually

⁶The term *corpus* denotes the entire document collection, i.e., the infobase.

a very high dimensionality in practical applications. Thus, a document x is represented by its vector $\mathbf{x} = \langle x_1, \dots, x_i, \dots, x_n \rangle$. x_i can be a boolean value (1 denoting the presence of term t_i , 0 its absence), or a function of the number of occurrences of the corresponding term t_i in the document. Often, the *tf-idf* (term frequency-inverse document frequency) is used, so that

$$x_i = f_{t_i} \log \frac{|D|}{F_{t_i}}$$

where f_{t_i} is the frequency of term t_i in document x , normalized with respect to the total term occurrences in the document; $|D|$ is the total number of documents and F_{t_i} is the number of documents containing t_i in the corpus.

In the vector space model, the similarity between a document and a query (or between any two documents in the corpus) can be formally defined in terms of the distance between the corresponding vectors. The most popular measure of similarity is the cosine measure

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$

which uses the cosine of the angle between the two vectors. The cosine measure is 1 when the two vectors are equal (the angle between them is zero degrees) and is 0 when the two vectors are independent (no terms in common). Other similarity measures were proposed. Among these, the Jaccard coefficient [150] for binary vectors

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|}$$

shows perhaps more clearly that the notion of similarity is actually based on the co-occurrence of the same terms in the two vectors/documents. Differently from the boolean model, the result list can be ordered by “relevance”, i.e., by similarity of retrieved documents with respect to the query.

The retrieval effectiveness of both the boolean and the vector space model depends heavily on the definition of *term*. Very often, a term is just a string of characters in the document between predefined separators. In this case, the dramatic semantic gap between the system model and the user model is quite evident. The user wants to retrieve documents containing specific concepts, such as “architecture in the Renaissance”, whereas the system works at a string level, so that the system might retrieve *architecture* but not *architectures*, *church*, or *15th century buildings*. Several approaches attempt to bridge this gap: grammatical variants can be conflated into a single form, either by recording the word stem [215] or some normal form; thesauri such as WordNet [100] can be used to expand queries with synonyms or related words. Still, it is very difficult to guarantee an accurate match between the query as intended by the user and the query as understood by the system. In the example at hand, Leon Battista Alberti’s *De Re Aedificatoria*, the most important book on Renaissance architecture, does not contain the term *Renaissance* simply because

it was coined at a later time. From this point of view, one of the most important differences between information retrieval and dynamic taxonomies, as well as one of main motivations of dynamic taxonomies [236], is that dynamic taxonomies are defined on concepts.

In order to assess the effectiveness of an information retrieval system, two measures are used: *precision* and *recall*:

$$Precision = \frac{|\{retrieved_documents\} \cap \{relevant_documents\}|}{|\{retrieved_documents\}|}$$

$$Recall = \frac{|\{retrieved_documents\} \cap \{relevant_documents\}|}{|\{relevant_documents\}|}$$

These measures are based on a subjective expert evaluation of the relevant documents for a given user query.⁷ Precision measures the absence of noise (irrelevant documents) in the result, while recall measures the completeness of the result. Ideally, $precision = recall = 1.0$, but empirical evidence [44] shows that precision tends to be good (in the order of .80), while recall is usually quite poor (around .20, in the reported study). Consequently, the central problem of information retrieval is the improvement of recall, without reducing precision.

3.1.2.1 Extensions

Once a measure of similarity among documents is defined, it can be used to cluster documents in such a way that similar documents are grouped together. There are many different clustering techniques (see [152] for a review) and they have been applied to different domains. The basic idea in clustering is to group documents in such a way that the similarity of any two documents a and b in a group C is larger than the maximum similarity between one of a or b and document d not in C . That is, documents in a cluster are more similar among them than with documents not in the cluster.

The barycenter of a cluster,⁸ called *centroid* of the cluster, is often used as a representative of the cluster itself. In addition to flat clustering, hierarchical clustering can be implemented by clustering in step i the centroids of the clusters computed in step $i - 1$. The hierarchical structures produced by hierarchical clustering are not taxonomies [236]. Clustering, in fact, is based on co-occurring terms. Consider the concept crime, which is an abstraction of several possible types of crime, e.g., crimes against property (further specialized into theft, fraud, ...), crimes against persons (murder, assault, rape, ...), etc. This natural taxonomy would not be produced by clustering because the resulting abstraction will probably contain clusters

⁷These measures are appropriate for the exact match retrieval models (e.g., the boolean model). For best match retrieval models (like the vector space model) other measures are used (e.g., Precision-Recall curves [178]) which measure the quality of ranking.

⁸The concept of a barycenter requires an underlying vector-space model.

such as (rape, murder), (rape, assault) since these crimes tend to co-occur and, almost certainly, no cluster (theft, fraud) since these do not co-occur. Consequently, the taxonomy tends to be based on the co-occurrence of concepts, rather than on an IS-A relationship, which the user implicitly assumes in a taxonomy.

Clustering can be used to speed up execution because of the cluster hypothesis: if a document d in a cluster C is relevant for a query, all the other documents in C , being similar to d , are also relevant for that query. Consequently, the system need not to compute the similarity of each document with the query, but can simply compute the similarity of the query with each centroid and subsequently expand the relevant centroids into their clusters.⁹

For the purpose of our discussion, clustering is interesting because it can be used to provide exploration capabilities to information retrieval systems. The Scatter-Gather algorithm [77] implements a SAES structure through dynamic (re)clustering in the following way. Initially, the user is presented with the topmost clusters for the corpus. Upon the selection of one or more specific clusters (taken as a union), which become the focus, the system will dynamically recluster (scatter) the selected documents, producing more refined clusters, and so on. Since the computational cost of reclustering is extremely high and unsuitable for real-life information bases, this process can be optimized [76] by precomputing a hierarchy of clusters so that the simple explosion of a cluster does not require any reclustering since its children were precomputed. Scatter (i.e., dynamic reclustering) is used when the user takes the union of two or more clusters at the same level. In this case, the centroids of the children clusters (rather than the actual documents) are reclustered, in order to achieve a reasonable performance.

When compared with dynamic taxonomies, however, Scatter-Gather appears inherently less effective [236]. First, hierarchical clustering does not really implement a taxonomy, as we noted above, so that the organization implemented is not the ‘natural’ IS-A organization that a user expects. Second, hierarchical clustering is monodimensional, i.e., a specific document belongs to a single cluster, whereas dynamic taxonomies are multidimensional.¹⁰ As we show in the following section, multidimensional access is much more selective than monodimensional access. In fact, the only way of changing the predefined hierarchical structure is by taking the union of two or more clusters: this may change the hierarchical structure, but obviously increases the size of the result. Third, clusters need to be identified by a clearly understandable label: as the example in [236] clearly shows, this is quite difficult, so that user orientation is quite difficult. Finally, clustering techniques are inherently language-dependent whereas dynamic taxonomies are language-independent.

Although Scatter-Gather cannot be applied to large corpora, it inspired a simplified strategy that is becoming popular [205], and is used, among others, by Vivisimo

⁹Search performance was one of the early motivations for clustering but it is no longer valid, as brute force strategies that would benefit are no longer used in practice.

¹⁰Some hierarchical clustering algorithms (e.g., the Suffix Tree Clustering algorithm [330], see also Chap. 8) produce overlapping clusters and an object can belong to more than one cluster.

The screenshot shows the Clusty search interface. At the top, there is a navigation bar with links for 'web', 'news', 'images', 'wikipedia', 'blogs', 'jobs', and 'more'. A search bar contains the query 'dynamic taxonomies'. Below the search bar, the Clusty logo is visible. The main content area is divided into two sections. On the left, under the 'clusters' tab, there is a list of 'All Results (170)' with various clusters such as 'Workshop on Dynamic Taxonomies (46)', 'Knowledge (23)', 'Information Bases (15)', 'Science (12)', 'Giovanni Maria Sacco (10)', 'Semantic Web (10)', 'Discovery (10)', 'Classification (10)', 'Practices, Research (6)', and 'Retrieving Information (8)'. A 'remix' button is also present. On the right, the search results are displayed, showing the top 170 results of at least 807 retrieved for the query. The first four results are listed, each with a title, a brief description, and a URL. The first result is 'Dynamic Taxonomies -- Homepage', the second is 'SIG-FIND -- Special Interest and Research Group on Dynamic Taxonomies', the third is 'FIND'08 -- 2nd Int. Workshop on Dynamic Taxonomies and Faceted', and the fourth is 'Knowledge Processors'.

Fig. 3.3 Results for web full-text search “dynamic taxonomies”; clusters to be used for refinement are on the left

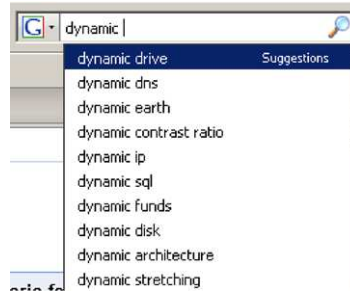
[314]. The idea is to cluster the result of an information retrieval query, not considering the entire document but only the small portion containing the query terms that search engines such as Google show (usually called *snippets*). For large result sets, only part of the results can be used for clustering; this improves performance, but obviously decreases the quality of clustering. A sample interaction is shown in Fig. 3.3.¹¹

With respect to our framework, clustering represents a SAES with restrictions. In fact, the interaction is not self-sufficient, because it starts with a full-text query, and the organization, although hierarchical, is not taxonomical for the reasons explained above. So, the benefits are really in improving precision by allowing the user to focus on interesting subsets of the result, rather than in improving the recall. Obviously the same arguments against clustering in Scatter–Gather [236] hold in this case as well (see also the later paper by Hearst [130]).

Another improvement over plain information retrieval is the suggestion of *related queries* in search engines such as Google [148] (see also [30]). When the user enters a search term, the system automatically displays popular queries that include that search term either as a whole or as a prefix. An example is shown in Fig. 3.4. With respect to SAES, this suggestion applies only to the query structure, supplying additional selectors with respect to the query term (selector) that the user has

¹¹Result clustering can be combined with DTs (see Sect. 8.3.1).

Fig. 3.4 Google toolbar suggestions for the term “dynamic”



to manually enter without assistance¹² anyway. So the impact of this technique, as well as in Vivisimo, is really in improving the precision of the query rather than its recall.

Finally, both query expansion (based either on local or on global analysis) and relevance feedback help users to improve their query [28]. The first method (especially when a thesaurus is used for expansion) improves query recall by considering additional terms, while relevance feedback exploits user feedback (e.g., documents approved or discarded) to adjust relevance weights and consequently improve precision. Neither method offers exploratory capabilities.

3.2 Static Taxonomies

In classical taxonomies, such as Linnaeus's,¹³ an object (e.g., *the dog*) is classified under one and only one concept. The taxonomy can be seen as an efficient encoding of properties of objects, which can be recovered by following the path from the concept under which the object is classified to the root concept: thus, we can state that a dog is a mammalian, has a spinal cord, is an animal.

More interestingly in our context, taxonomies systematically organize data in order to make the search for specific objects¹⁴ more efficient. The user starts from the root concept and iteratively discriminates among son concepts, in order to find the appropriate one. Each time a concept is selected for expansion, the total number of objects to be considered is reduced because the objects classified under discarded concepts need not be considered. Thus, the user iteratively reduces the number of documents to be manually inspected, by descending the taxonomy.

In classical, static monodimensional taxonomies, the maximum reduction occurs at terminal concepts: a terminal concept is not further specializable, and therefore all the documents classified under it must be manually inspected. In dynamic taxonomies, however, a terminal concept C can be reduced by another concept C'

¹²Other than the autocomplete mechanism.

¹³Carl Nilsson Linnaeus, *Species Plantarum*, 1753.

¹⁴As we will note in the Sect. 3.3, a taxonomy used for search defines a decision tree.

related to C through the extensional inference rule, so that a larger reduction is expected.

We are interested here in the retrieval effectiveness of different taxonomic approaches [247]. Retrieval effectiveness will be measured by the *maximum resolution of the taxonomy* (MR) which represents the average minimum number of documents to be manually inspected, after a refinement through operations on the taxonomy only.¹⁵ The larger MR is, the less effective the access through the taxonomy is as far as discriminative power is concerned, and the more work the user has to do. For concreteness, we will assume a target $MR = 10$. This is an approximation of the maximum number that most users will be willing to manually inspect.

We will analyze the following approaches:

- *Static monodimensional taxonomies*. It is the most common browsing paradigm, and it is used, among others, by Yahoo!
- *Multidimensional taxonomies with no concept composition capabilities*. This paradigm simply extends monodimensional taxonomies by allowing documents to be classified under several concepts, but retains the visual framework of conventional taxonomies, i.e., level-by-level expansion, so that concept composition is not available to end-users. This approach was used in the past by Microsoft Knowledge Manager v. 1.0¹⁶ among others.
- *Multidimensional taxonomies with concept composition capabilities*. This paradigm is implemented by dynamic taxonomies, and by all the systems that support boolean queries on taxonomic metadata. The maximum resolution is the same for the two approaches. The fundamental difference is that dynamic taxonomies are SAESs and integrate both concept composition and feedback in the same visual framework. Boolean queries on metadata do not provide any feedback to the user who therefore has to guess which concepts to compose and suffer from most of the user interaction problems found in text retrieval systems.

In our analysis of the maximum resolution, an information base of D objects and a taxonomy with T terminal concepts with objects classified under terminal concepts only, will be assumed. In addition, a uniform classification probability distribution and the independence of terminal concepts will also be assumed. These latter assumptions are used to simplify the following analysis but are acknowledged not to be realistic (violation of these hypotheses is discussed in Sect. 3.2.6). For this reason, the analysis validation reported in Sect. 3.2.7 is especially important, since it is performed on a real corpus, for which these assumptions are not verified.

¹⁵Since we are interested in the average *minimum* number of documents, our analysis will consider terminal concepts, rather than the higher levels of the taxonomy.

¹⁶*Circa* 1999.

3.2.1 Static Monodimensional Taxonomies

In static monodimensional taxonomies no concept composition is possible, except trivial ones: for any two concepts C and C' , $C \cap C' = \emptyset$ unless C' is a descendant or an ancestor of C , in which case, trivially, $C \cap C' = C'$.¹⁷ The zoom operation on a concept C trivially preserves only the descendants and the ancestors of C . In monodimensional taxonomies, terminal concepts cannot be further refined, and consequently the entire set of objects classified under a terminal concept must be manually inspected. Under our assumptions of a uniform distribution of documents, $MR = D/T$.

With a static monodimensional taxonomy, a target of $MR = 10$ requires $T = D/10$, i.e., that the number of terminal concepts be just one order of magnitude smaller than the size of the information base. This implies that very large taxonomies are required for real-life information bases and that the taxonomy must grow as the corpus grows and therefore that terminal concepts must be indefinitely specializable, which is a condition very difficult to satisfy.

3.2.2 Static Multidimensional Taxonomies with no Concept Composition Capabilities

This approach extends monodimensional taxonomies in a straightforward way, by allowing objects to be classified under several concepts while retaining the same access paradigm as static monodimensional taxonomies: the user navigates the taxonomy tree in the same top-down way as in monodimensional taxonomies.

Access through the taxonomy suffers from the same problems as in static monodimensional taxonomies. The most important problem is that concept composition cannot be exploited because, once a branch of the taxonomy is chosen, subsequent refining can only involve the descendants of the selected branch. All the different branches of a taxonomy are independent and all the relationships among concepts, other than the subsumptions explicitly defined in the taxonomy, are completely lost. Consequently, terminal concepts cannot be further refined, and the entire set of objects classified under a terminal concept must be manually inspected.

If we extend static monodimensional taxonomies to a multidimensional classification, the maximum resolution becomes considerably worse. In fact, if each object is classified under j terminal concepts, the resulting MR is equal to the MR for the monodimensional classification of jD objects, i.e., $MR = jD/T$. In order to reach our objective, we must have $T = jD/10$, which means that the required number of terminal concepts may well exceed the size of the information base when a non-trivial number of dimensions is used.

¹⁷Even if C and C' are incomparable (none of them is descendant of the other), it could still hold $C \cap C' \neq \emptyset$, if the taxonomy is a DAG and C and C' have a common descendant.

3.2.3 *Multidimensional Taxonomies with Concept Composition Capabilities*

This approach supports the composition of concepts through boolean operators, which can be translated into the corresponding set theoretic operations on the deep extension of the concepts involved. Concept composition can be supported ‘syntactically’, in the sense that the user enters a boolean query on concepts in the taxonomy, as in an early implementation of `factiva.com`. Alternatively, a dynamic taxonomy can be used both to compose concepts and to give the user a systematic summary of the result of such composition. Such summaries include all the concepts related to the result, which can be used in further composition, and therefore guide the user to reach his goal.

We are concerned here with composition in *and* (which results in the intersection of the deep extension of the concepts involved) since it restricts the information base. As a consequence of concept composition, a leaf or terminal concept L can be composed with another concept C (possibly a terminal concept as well), so that L is further “specializable” even if it is a terminal. One of the differences between the two approaches is that in the former ‘syntactic’ approach L and C can result in an empty set, whereas in dynamic taxonomies this cannot occur, by construction.

Since terminal concepts can be refined by other terminal concepts, the maximum resolution MR depends on the most restrictive combination of terminal concepts which can be performed on the dynamic taxonomy. If each object is classified under j terminal concepts, the maximum number of terminal concepts which can be intersected is j .¹⁸ Clearly, the intersection of k ($k < j$) terminal concepts is no smaller than the intersection of j terminal concepts, and the intersection of j concepts, some of which are non-terminal, is likewise no smaller.

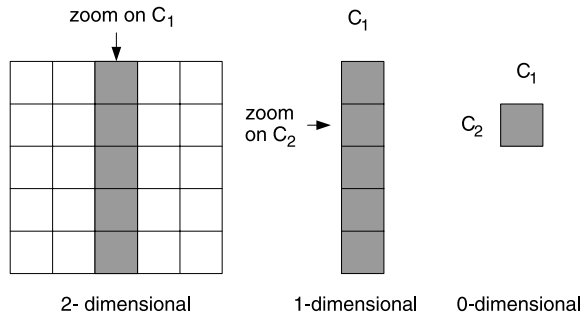
In order to compute the maximum resolution of a dynamic taxonomy, in which each object is classified under j terminal concepts, we will consider the two scenarios described below. In practice, real scenarios are usually a mix of these scenarios and, quite often, the number of concepts under which an object is classified is not fixed, but varies within the corpus.

3.2.3.1 *Faceted Classifications*

In the first scenario, each object is described by a set of j facets, each represented by an independent taxonomy. As an example, consider a digital camera which is described by features such as Price, Weight, Resolution, etc. The values of each feature can be arranged taxonomically. In this case, we assume that the set of terminal concepts in the dynamic taxonomy is partitioned into j subsets of the same size.

¹⁸The intersection of more than j terminal concepts is guaranteed to produce a null result.

Fig. 3.5 Two zoom operations on a two-facet taxonomy/two-dimensional hypercube



Under this assumption, each partition has T/j ($T/j \geq 2$) terminal concepts. Each object d is classified under one and only one leaf concept L in each partition. In addition, we assume that the probability of classifying d under L is uniform for each partition.

In this case, MR can be computed as $MR = D(j/T)^j$, $T \geq 2j$. As a geometric interpretation, note that the present scenario corresponds to a j -dimensional hypercube, in which each side has T/j elements. By subsequent zooming, we iteratively remove (fix) a dimension to the hypercube, until, after j zoom operations, we arrive at the 0-dimensional hypercube, i.e., the cell. The cell represents, in our context, the objects to be inspected. Figure 3.5 shows two zoom operations on an infobase described by a two facet taxonomy, or, equivalently for our current purposes, by a 2-dimensional hypercube. The first zoom operation on a specific concept C_1 slices the 2-dimensional hypercube according to C_1 , and produces a 1-dimensional hypercube. The second zoom operation operates on the other facet/dimension, and slices a 1-dimensional hypercube, producing a 0-dimensional hypercube, i.e., a cell. In OLAP the cell contains the fact variables, whereas in DTs it contains the objects to be manually inspected by the user. In this geometric interpretation, therefore, MR is the average number of objects in cells obtained by zooms on terminal concepts only.

Incidentally, the geometric interpretation shows that there is a strong correlation between dynamic taxonomies and OLAP techniques based on hypercubes, and that, as mentioned before, the browsing system of dynamic taxonomies can be useful for interactive exploration in the context of OLAP as well [37, 247].

3.2.3.2 Unrestricted Multidimensional Classifications

The second scenario, which is common in textual applications, has a single taxonomy (e.g., Topics), j terminal concepts are chosen to classify each object, and the only constraint is that a specific concept C can be chosen only once. In this case, we can compute MR as a function of the number S_j of the sets generated by the intersection of j distinct terminal concepts, i.e., $MR = jD/S_j$. The maximum number

of different intersections S_j is equal to

$$S_j = \binom{T}{j} = \prod_{1 \leq k \leq j} \left(\frac{T+1-k}{k} \right)$$

which, assuming $T \gg j$, can be grossly approximated by $S_j = T^j$. Consequently, $MR = jD/T^j$.

3.2.4 A Comparison

A dynamic taxonomy has an MR which is at least $(T/j)^{j-1}$ better than a static multidimensional taxonomy. The analysis above shows that static taxonomies cannot provide efficient access to large information bases. With a static monodimensional taxonomy, our target MR ($MR \leq 10$) requires $D = T/10$, i.e., the number of terminal concepts must be just one order of magnitude smaller than the size of the information base. Static multidimensional taxonomies have a worse MR than monodimensional taxonomies. In this case, $D = jT/10$, which means that the required number of terminal concepts may well exceed the size of the information base when a non-trivial number of dimensions is used.

The situation immediately becomes manageable with a j -dimensional dynamic taxonomy. In fact, the target MR requires $D = 10(T/j)^j$ or $D = 10T^j/j$ for scenario 1 and 2, respectively. This means that a compact taxonomy of 1,000 terminal concepts and a classification scheme in which each object is classified under 10 terminal concepts are sufficient to reach $MR = 10$ for information bases with as many as 10^{21} (scenario 1) or 10^{30} (scenario 2) objects. A static taxonomy with the same number of terminal concepts reaches $MR = 10$ for infobases no larger than 10^4 objects.

It can be interesting to perform analysis in the reverse direction, i.e., to determine the minimum number of terminal concepts (MT) required by a dynamic taxonomy with a faceted classification vs. a static taxonomy, as a function of the infobase under the constraint of $MR = 10$. In the case of a static taxonomy $MT = D/10$. In the case of a faceted dynamic taxonomy, assuming a complete tree taxonomy with a constant fanout of f ,

$$MT = fj \quad \text{such that } D/(f^j) = 10 \text{ and } fj \text{ is minimum.}$$

Since fj is minimized by the minimum fanout 2, we have $MT = 2j$ s.t. $2^j = D/10$, that is, $MT = 2 \log_2(D/10)$. Assuming now, for concreteness, an infobase of 1 billion objects, a static taxonomy requires 100 million terminal concepts, whereas the most compact faceted dynamic taxonomy requires 60 terminal concepts. The minimum number of terminals is a theoretical quantity because a classification by binary facets might not be useful, or even possible, in practice. However, it provides a clear perception of the difference in expressive power between dynamic taxonomies and static taxonomies.

Because of the high reduction factor obtained by dynamic taxonomies, zooms on less than j terminal concepts are usually sufficient. We call $MR(i)$ the maximum resolution deriving by the intersection of i terminal concepts, $i \leq j$:

$$MR(i) = \begin{cases} j^i D/T^i & \text{in scenario 1,} \\ j D/T^i & \text{in scenario 2.} \end{cases}$$

The maximum information base size D with $MR(i) = 10$, under the assumptions above, varies from $D = 10^{2i+1}$ to $D = 10^{3i}$ for scenario 1 and 2, respectively. This means that the composition of three concepts is sufficient for information bases with 10 million to 1 billion objects.

Finally, it could be contended that the same reducing effect of dynamic taxonomies can be achieved by representing all the possible concept intersections¹⁹ as explicit concepts in the taxonomy. However, this strategy results in an exponential growth of the taxonomy. Consider a faceted taxonomy on j facets and T/j terminals per facet: the number of different facet combinations which can be obtained at the terminal level is $(T/j)^j$. An equivalent monodimensional taxonomy would require these many terminals. Assuming $j = 10$ and $T = 1000$, the equivalent monodimensional taxonomy would require 100^{10} terminals, which is obviously impractical.²⁰ Dynamic taxonomies can synthesize all the possible non-empty concept intersections. Although such an exhaustive synthesis would require an exponential time, concept intersections are generated on request, and in general, only an extremely small subset of all the possible intersections is generated.

3.2.5 Taxonomy Pruning in Dynamic Taxonomies

One of the most important features of dynamic taxonomies is the production of conceptual summaries of arbitrary sets of objects from the universe. Conceptual summaries are extremely important to guide the user towards his goal, since they show the user all and only those concepts which can be used for further refinements. These summaries are computed on the basis of the actual classification only, according to the extensional inference rule. The usual way of displaying such summaries is through a reduced taxonomy, i.e., the original taxonomy from which all the concepts which are not related to the current focus are pruned.

Although concept pruning is not required per se in order to guarantee a good convergence, an aggressive pruning simplifies user interaction because it reduces the number of selectors the user has to consider. For this reason, it is interesting to analyze the expected pruning in the taxonomy. We keep our simplifying assumptions of a uniform classification distribution and concept independence, and we want to

¹⁹Only concept intersections are considered here, whereas dynamic taxonomies allow all the boolean operations on concepts. Obviously, this considerably reduces expressivity.

²⁰Many of these concepts will be meaningless (see Sect. 6).

estimate the number of unique terminal concepts under which the s objects in set S that we want to conceptually summarize are classified.

This problem can be reduced to the problem of estimating the number of block accesses in database systems, for which well-known solutions exist. The block accesses estimation problem is formulated as follows:

given a file of P pages with a blocksize (number of records per page) equal to b , we want to estimate the number of unique pages U which are selected when k random records must be accessed.

The number of block accesses is estimated by Yao's function [325], which is computationally expensive but can be approximated by Cardenas formula [58]

$$U = P(1 - (1 - 1/P)^k)$$

In order to reduce our problem for unrestricted multidimensional classifications to this one, we will draw the following equivalences:

1. the number of pages P in the file are equivalent to the T terminal concepts in the taxonomy
2. the blocksize b (which is ignored by Cardenas approximation) is given by jD/T , i.e., by the average number of documents which are classified under each terminal
3. the k random records are equivalent to the number of classification descriptors in the set S which we summarize: i.e., to $j|S|$.

Since we are interested in the pruning ratio, we will concentrate on the portion $(1 - 1/P)^k$ of Cardenas formula, which becomes $(1 - 1/T)^{j|S|}$ for our problem. Now, we investigate the pruning ratio p as a function of the number i of concept compositions we perform:

$$p = \left(1 - \frac{1}{T}\right)^{j(jD/T)^i}$$

In order to get a feeling of the estimated pruning, consider an information base of 100,000 documents classified under 10 dimensions through a taxonomy of 1000 terminal concepts. While the amount of pruning is negligible for a single zoom, focusing on two concepts prunes an estimated 90% of the total terminal concepts.

The faceted classification case is similar but the number of pages P is no longer equivalent to T , but to T/j , because all the facets are independent. The estimation formula becomes:

$$p = \left(1 - \frac{j}{T}\right)^{j^i(jD/T)^i}$$

and, under the same assumptions, the composition of two concepts prunes an estimated 36% of total concepts.

3.2.6 *Relaxing the Assumptions*

The assumptions used to simplify the analysis are not valid in general. In many practical situations, we will expect the classification probability distribution to deviate from uniformity: for example, a small number of concepts (e.g., the main focus of the infobase) will be quite popular, while a large number of concepts will occur rarely if ever. In general we expect distributions such as Zipf [337] or 80–20 [137] to occur in practice. The assumption on concept independence may also be violated. Consider a concept such as “aerospace” and a location index. We know that aerospace is closely correlated to developed countries such as USA, Europe, Russia, etc. while other countries, such as African ones, will have no occurrences of this concept.

If we consider the impact of violations of these assumptions, it is obvious that the critical assumption is the independence of concepts. Assume that concepts A and B are totally correlated: in this case, obviously, A will be useless to refine B and vice versa, since focusing on a concept will preserve all the occurrences in the other concept. Such a complete correlation is unlikely to occur in practice, because it implies that the two concepts are really aliases. However, we can expect “local” deviations from independence, i.e., some correlation among groups of concepts, so that the actual discriminative power of a dynamic taxonomy will be lower than the estimation we provided.

The assumption on uniform classification distribution is less critical, if the independence assumption holds. Assume that we have a faceted classification, and that, for each facet, the classification distribution is Zipfian. In a naïve static multi-dimensional classification this is likely to be a problem: while the vast majority of concepts will select a number of documents lower than the average, a small number of concepts will select a very large number of documents. A user interested in high-frequency concepts will be forced to inspect a significant percentage of the infobase.

If the independence assumption holds, however, the first zoom operation will considerably reduce the maximum number of documents to be inspected. Let us consider the highest-frequency concept in a facet. Its probability, assuming a Zipfian distribution and T/j terminals per facet, is $1/H_{T/j}$, where $H_{T/j} \approx \ln(T/j) + \gamma$ is the (T/j) th harmonic number and $\gamma \approx 0.67$ is Euler’s constant. The intersection of the highest frequency concept for i facets will select a number of objects equal to $D/(\ln(T/j) + \gamma)^i$.

In our example, the maximum number of documents selected by a concept is 18,975, but the combination of three concepts reduces this quantity to 683. Thus, if the user intersects the highest frequency concepts he will experience a slower convergence: each high-frequency concept added to the intersection reduces the selected documents by a factor equal to $(\ln(T/j) + \gamma)$ instead of T/j . However this slower convergence occurs only when the user continues to select the highest-frequency concept for each partition: as soon as the user selects a low-frequency concept the system will converge faster than under the uniform distribution assumption.

3.2.7 Experimental Results

In order to assess the behavior of dynamic taxonomies in the large, we worked on three different collections of documents. The first two collections include a collection of Italian news from 1992 and 1993, respectively. The third collection is the union of the first two. Although the collections we consider are very small when compared to the infobases managed by web search engines, they are real-world infobases with a general focus and are sufficiently large to insure the statistical significance of our experiments.

All the collections were automatically reclassified from an existing classification based on a controlled set of keywords, and are characterized by a multidimensional classification which is a mix between the two scenarios described in the previous section: in fact, each article is characterized as in Scenario 1 by the values of several independent features, such as Location, Type of Article (e.g., interview, editorial), etc. In addition, the “information content” of the article is described, as in Scenario 2, by several concepts in the Subject taxonomy.

As we expected, the classification distribution for articles is not uniform. As an example, the reader is referred to Fig. 3.6, which reports the counters for articles organized by location. The distribution is probably Zipfian since we have more than 25,000 articles related to Northern Italy, whereas the vast majority of countries (e.g., Haiti, Honduras, etc.) has one or no document associated. This is entirely to be expected since the focus is on internal news and on such international news likely to affect or interest the Italian reader. In addition to deviations from uniformity,

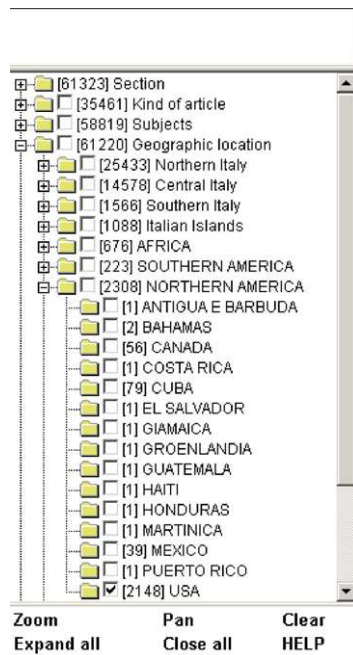


Fig. 3.6 Number of articles per country

Table 3.1 Collection statistics

Collection	Total docs	Size	Concepts	Terminal concepts	Avg. docs per terminal	Avg. terminals per doc
1992	57590	179.0 MB	1106	1021	609.41	10.80
1993	56452	180.6 MB	1109	1024	639.05	11.59
1992+1993	114042	359.6 MB	1109	1024	1246.22	11.19

Table 3.2 Experimental results

Collection	Avg. result size	Estimated MR(2) range	
1992	1.87	0.6–6.44	
1993	2.47	0.72–7.23	
1992+1993	5.25	1.22–13.62	

Collection	Number of nonempty sets	Avg. size of nonempty sets	Nonempty sets with size > 10
1992	1415	13.22	270
1993	1674	14.77	339
1992+1993	1897	27.68	491

the experimental corpora also violates the assumptions we made on concept independence. As an example, the concept “movies” (as seen from Italy) has a higher correlation with Italy, USA, France than it has with Haiti, Honduras, etc.

Table 3.1 reports the principal statistics for the three collections. The columns Concepts and Terminal concepts count non-empty concepts only, i.e. concepts under which at least one document is classified. About 1000 concepts were used to classify the collections. The average concepts per document indicates under how many different terminal concepts each document is classified: about 11 concepts were used on the average to describe a single document. The average document per terminal concept measures how many documents are retrieved by simply exploding the taxonomy to a terminal concept, i.e., the *MR* for a conventional multidimensional taxonomy. As expected, the number of documents to be manually inspected is very large and consequently the conventional taxonomy is not useful for retrieval.

The benefits in terms of information thinning are reported in Table 3.2. For each corpus, we computed 10,000 intersections of two concepts conservatively selected at any level in the tree, rather than only at the terminal level which is obviously much more discriminative. We note that at worst only 491 intersections out of 1897 non-empty intersections (about 25%) resulted in sets with a size larger than 10, for which additional intersections are required. In addition, about 80% of the intersections resulted in an empty set, which indicates a quite effective pruning of concepts in the computation of reduced trees, as expected by our analysis.

The observed measures match our estimates²¹ and show that the deviations from uniformity and from concept independence that occur in practice do not degrade the expected performance of dynamic taxonomies in any significant way. Interestingly, the average size of non-empty sets seems to be twice the estimated *MR* for a faceted classification. Additional experiments are required to verify if this relation holds for different infobases.

The advantages of dynamic taxonomies over conventional access methods are *dramatic*: whereas a static taxonomy would require the user to inspect result lists of several hundreds documents, a dynamic taxonomy requires only one additional zoom operation to produce an average result size of 13 to 27 documents.

In conclusion, we have shown that static taxonomies, and especially those which support a multidimensional classification, are not appropriate for non-trivial information bases. In static taxonomies, the finest specialization level (terminal concepts) cannot be further refined. If we place a reasonable bound on the number of documents to be manually inspected, the number of terminal concepts required becomes so large to be unfeasible. Simple multidimensional extensions or naïve faceted organizations make these problems even worse.

On the other hand, dynamic taxonomies are inherently extremely effective for reducing very large information bases. A very limited number of concept intersections is sufficient to produce result sets which are no larger than 10, on the average, even for very large information bases. We conclude that dynamic taxonomies are a requirement for effective taxonomic retrieval. If they are not used, taxonomic retrieval is impossible *per se*, due to the large size of result sets, and the user has to resort to other retrieval techniques (database queries, text retrieval queries, etc.) to thin information out.

3.3 Decision Trees

Moret [194] defines a decision tree as a device to identify objects, in the following way:

given a boolean function on n variables (tests), a decision tree can be regarded as a deterministic algorithm for deciding which variable to test next, based on the previously tested variables and the results of their evaluation, until the function's value can be determined. Decision trees are rooted, ordered and labeled trees.

Hence, in information access, the user of a decision tree is presented with a fixed, predefined decision structure which he has to explore in the order defined by the designer. In most implementations, the decision tree has to be followed in its entirety up to a leaf, with no indication of candidate objects after each test.

The main differences between decision trees and dynamic taxonomies are in object access and in the creation/evolution of the decision structures.

²¹The estimated range reports unrestricted classification and faceted classification *MR* estimates, respectively.

Using a static monodimensional taxonomy for object access is obviously equivalent to a decision tree with the same hierarchical structure of the taxonomy (and the reverse holds). At each stage, the test selects a single son among the set of available sons of the current node. In dynamic taxonomies, instead, there is no notion of order among features and the user can freely focus on any feature he finds relevant, in order to restrict the portion of the information base to be considered and to determine all the other features which can be tested, in a totally dynamic way. In fact, by selecting an arbitrary sequence of zoom operations, the user dynamically synthesizes a specific decision tree out of all the possible ones.

A predefined order is inappropriate for most explorative tasks, because it forces a predefined order on user perspectives, and consequently constrains exploration. For example, consider the typical object-seeking task, the selection of a product in e-commerce. A decision tree implies a predefined order on n features. For example, for a digital camera, the predefined order could be Price > Resolution > Zoom > Weight. Dynamic taxonomies, instead, offer an unconstrained exploration which results in the dynamic synthesis of any of the $n!$ possible orders on n features.

Although in most cases the cost of a test is a constant, there are a number of important applications in which this is not true. For example, consider medical diagnosis. Here a decision tree *à la* Oncodoc [267] requires the tests to be performed in a rigid, predefined order. However, the cost of a specific test can depend on the capabilities of the medical unit where the user is located: thus, a test which can be performed in a few hours in a central hospital, might require the transportation of a patient from a decentralized unit. This indicates that access through a decision tree can be significantly suboptimal in some cases. Optimization of decision trees with respect to the cost of tests is known to be an NP-complete problem [194].

In dynamic taxonomies, total freedom in selecting the sequence of tests allows the user to perform low-cost tests first, and adapt to his specific environment. In addition, at each step, the user has a complete indication of all the objects which satisfied the tests performed so far. This is especially important, because the user can often rely on his knowledge/memory to select the right object in a relatively small set of candidates, instead of applying additional tests. Finally, dynamic taxonomies support the notion of a taxonomic organization of features, which guides users in easily locating available tests and allows them to select the appropriate level of abstraction for their inquiry.

From the user interaction point of view, selection by decision trees is the analog of interacting through a predefined modal dialog in which the system asks the questions: this is a system-centric master-slave architecture. Access through dynamic taxonomies guides the user to reach his goal by simply hiding selections and features which do not apply in the current state. As such, interaction is modeless and the system is perceived as an assistant rather than as a master.

As regards information classification, the notion of order is required in decision trees in order to select a specific tree among all the trees equivalent to the given boolean function. As we mentioned above, selecting an order for tests is critical for two reasons. First, different tests may have different costs: we want to anticipate low-cost tests and defer high-cost test, in order to minimize the total system costs.

Second, we might want to minimize the average or the maximum number of tests required. Decision trees are usually incomplete binary trees, so that very skewed trees can arise in practice. In addition, the probability of selecting a specific information object is usually not uniform: it makes sense to select a tree which minimizes the number of tests for the most frequently occurring pathologies. All these considerations call for the optimization of decision trees: but, as we said, this problem is known to be NP-complete [194].

The construction of a decision tree for m information objects requires identifying n features which can be used to *discriminate* among objects, and then use these features to construct the decision structure. This approach has two undesirable side effects. First, the addition of a new object requires the modification of the decision tree and in practical situations it may disrupt it. Second, and more importantly, since we use features to discriminate among objects, such discriminators are globally defined. That is, each object has to be compared against any other object in order to define appropriate discriminators: this means a quadratic complexity in the number of objects, so that applications of decision trees do not easily scale up.

In dynamic taxonomies, instead, m features are used to *describe* each object, in isolation. The classification of a single object o , i.e., the values of these features for o , depends on o only, which means that the complexity of the construction of the decision structure does not depend on the size of the information base.

3.4 Formal Concept Analysis

In this section, we review basic Formal Concept Analysis (FCA), following in part the introduction by Priss [217], and compare it to Dynamic Taxonomies (DT). A complete mathematical characterization of FCA can be found in Ganter and Wille [114]. Although FCA and DT are apparently two distinct approaches to information modeling and access, and they use a different terminology, they are closely related. We demonstrate in the following that they share a similar data model, a similar navigation space, and mostly differ in the way they interact with users. We think this sets a bridge so that each theory can gain from the other.

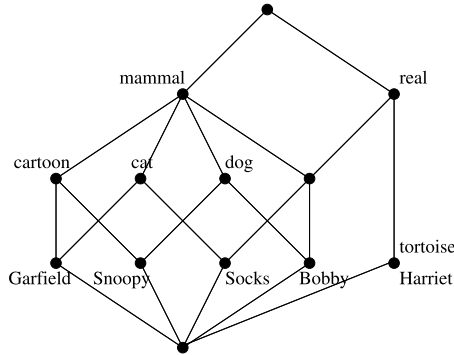
3.4.1 Data Model

FCA concerns with formal *objects* and formal *attributes*. Objects are usually real-world objects that are characterized by attributes. Thus, for example, digital cameras are objects for which attributes such as resolution, price are recorded. The sets of formal objects and formal attributes together with their relation to each other (e.g. what resolution a specific camera has), called the *incidence relation*, form a *formal context*, which can be represented by a cross table. A simple example, famous animals [217], is given in Table 3.3, where a \times in the cell corresponding to object o and attribute a means that a applies to o .

Table 3.3 A formal context of “famous animals”

	Cartoon	Real	Tortoise	Dog	Cat	Mammal
Garfield	×				×	×
Snoopy	×			×		×
Socks		×			×	×
Greyfriar’s Bobby		×		×		×
Harriet		×	×			

Fig. 3.7 Concept lattice for the context in Table 3.3



A *formal concept* c is then defined as a pair (O, A) where O is a set of objects, and A is a set of attributes, such that each attribute a in A applies to all objects o in O , and that each o in O has all the attributes a in A . A is called the *intent* of c , O the *extent* of c . A difference with concepts in dynamic taxonomies is that they are defined *a posteriori*, as a consequence of data, similarly to clustering approaches.

A duality called a *Galois connection* applies between extents and intents of concepts. A Galois connection implies that if one makes the sets of one type (e.g. the intension) larger, the corresponding sets of the other type (e.g. objects) become smaller or equal, and vice versa. Because of this property, a hierarchical ordering of concepts can be derived. In particular, a concept (O_1, A_1) is called a subconcept of (O_2, A_2) if $A_1 \supseteq A_2$ (and equivalently, $O_1 \subseteq O_2$). Such a hierarchical ordering of concepts (which defines a concept lattice) can be visualized by a line diagram such as the one in Fig. 3.7 [217].

All edges in the line diagram of a concept lattice represent subconcept–superconcept relations. The top (supremum) and bottom (infimum) concepts in a concept lattice are special. The top concept has all formal objects in its extension, and it is therefore the most general concept (in most cases, its intension is empty). The bottom concept has all formal attributes in its intension, and is empty if at least two attributes in the context are mutually exclusive, or if less than the total number of attributes is used to classify objects. Subconcept–superconcept relations are transitive, and therefore a concept is subconcept of any concept that can be reached by

traveling upwards from it. All the attributes in a formal concept are inherited by all its subconcepts.

In basic FCA, an attribute is a boolean value—either it applies or it does not. In most practical situations, however, attributes (e.g. prices, weights, brands) are multivalued. In order to deal with this problem, FCA introduces a mechanism (called *conceptual scaling*) in which an attribute can be exploded into its values.

Every taxonomy-based source can be translated into a formal context, and reciprocally, without any loss of information. The only condition is that a scale context be used in order to represent the taxonomy. In short, the translation can be summarized by the following correspondences.

DT	↔	FCA
object	↔	object
concept	↔	attribute
classification	↔	formal context
taxonomy	↔	scale context
infobase	↔	scaled context
focus	↔	formal concept

There is a clear correspondence between objects in the two models, as real-world entities, and between DT concepts and attributes, as elementary descriptors for those entities. The DT sentence “concepts are labels under which objects can be classified” can be transposed to FCA without difficulty: “attributes are labels under which objects can be classified”. However, note that DT concepts play a different role than FCA formal concepts. In the following, we distinguish between DT concepts and formal concepts, unless the context makes it clear which kind of concept is relevant. There is also an equivalence between the classification of objects under DT concepts and a formal context $K = (O, A, I)$, where the extent of an attribute corresponds to the shallow extension of the corresponding DT concept. Both are binary relations between objects and DT concepts/attributes. However, in DT, a second source of information comes from the taxonomy that organizes DT concepts in a subsumption ordering \leq . This is naturally represented in FCA by a scale context $S = (\mathcal{A}, \mathcal{A}, \leq)$, where \mathcal{A} is the set of all attributes [114]. There is a cross in the scale context at position (a_1, a_2) as soon as a_2 is an ancestor DT concept of a_1 . From there, FCA allows to combine the formal context and the scale context to form a scaled (formal) context $K_S = (O, A, I_S)$, where the extent of an attribute corresponds to the deep extension of the corresponding DT concept. The incidence relation of the scaled context is defined by

$$(o, b) \in I_S \iff \exists a \in \mathcal{A} : (o, a) \in I \wedge a \leq b.$$

Therefore, there is a total correspondence between the infobase of a dynamic taxonomy, and the combination of a scale context and a formal context. A first difference is that, in FCA, this is the scaled context that is effectively used to compute the concept lattice and support information access. However, in the scaled context, the taxonomy information is made implicit, so that when two attributes are ordered in the concept lattice, we cannot tell whether this is imposed by the taxonomy, or by

the classification of objects. We think it is important to distinguish the two cases, because they correspond to different levels of knowledge: general well-known knowledge in the first case (e.g., a taxonomy of continents and countries), and specific to-be-discovered knowledge in the second case (e.g., implications between countries and cultural features).

A second difference is in the correspondence between focus and formal concept. Both characterize a set of objects, but only FCA has the corresponding intent, and constrains the focus to be the extent of a formal concept. On one hand, the intent makes formal concepts more informative. On the other hand, the extent constraint limits the expressivity of the queries that can be used. The consequences of these differences on information access are discussed in detail in the following section.

3.4.2 Information Access

Formal Concept Analysis is applied to data analysis, information retrieval and exploration, and knowledge representation and discovery in domains as different as social sciences, software engineering and multimedia data [217]. We here focus on information retrieval and exploration. In fact, FCA has been recognized as a good solution for tightly combining querying and navigation [121], hence for supporting exploration. A query is defined as a set of attributes, and every query leads to the most general concept that has all the query attributes in its intent. The extent of this concept represents the set of answers to the query. The intent of this concept may have more attributes than the query. The additional attributes are *consequences* of the query attributes in the context, i.e., every object that has all query attributes also has these additional attributes as properties. From there, it is possible to navigate downward to reach more specific concepts (less answers), or upward to reach more general concepts (more answers). Compared to databases, the FCA approach has the advantage to guide users in their browsing, rather than just asking for queries. Compared to hierarchies (e.g., file systems), the FCA approach provides several paths to reach a set of objects because the order in which attributes are given is not constrained, and the navigation structure (the concept lattice) is automatically derived from data (the formal context).

A number of FCA systems, like the CEM (Conceptual Email Manager), display concept lattices like the one in Fig. 3.7. Navigation then simply consists in following edges from concepts to concepts. In order to allow for smaller and more readable diagrams, the formal context can be restricted to subsets of objects and attributes. However, because concept lattices are generally too large to be read entirely, most tools resort to local views over them [92, 181]. In the most complete form, every local view displays the intent and extent of the current concept, as well as children and parent concepts. These related concepts are represented only by the difference between their intent and the current concept intent, i.e., sets of attributes to be added or removed from the current intent. The computation of the concept lattice is exponential in the number of objects or attributes. This imposes strong limitations on

the size of formal contexts that can be browsed. A first solution is to extract a small formal context from a large information system by usual search techniques, and then apply FCA on it (e.g., CREDO for the Web [59], Docco for the desktop [72]). A second solution is to avoid the computation of the concept lattice, but to compute related concepts on demand [103, 171]. Thus, only the visited part of the concept lattice has to be computed.

The navigation space of DTs, when restricted to AND-composition, is exactly the concept lattice of FCA. If the zoom operation in DT is restricted to a AND-composition of concepts, every focus that can be reached is the extent of a formal concept, and reciprocally, every non-empty extent of a formal concept can be reached by a succession of zoom operations. Initially, the query is empty, and the focus is equal to the universe, i.e., all objects. This corresponds to the supremum formal concept. After k zoom operations on DT concepts a_1, \dots, a_k , the query is $q_k = a_1 \text{ AND } \dots \text{ AND } a_k$, and hence, the focus is $F_k = \bigcap_{i \in 1..k} \text{objects}(a_i)$. As every DT concept is an attribute of the scaled context, the focus F_k is necessarily the extent of a formal concept $c_k = (F_k, F'_k)$, where F'_k is the intent of c_k . From there, the dynamic taxonomy presents a conceptual summary of the focus by retaining only DT concepts that are extensionally related to it, i.e. any DT concept a such that $F_k \cap \text{objects}(a) \neq \emptyset$. In fact, every such DT concept a can be seen as a selector from the focus F_k to the focus $F_{k+1} = F_k \cap a$, corresponding to the query $q_k \text{ AND } a$. Of course, every focus F_{k+1} is again the extent of a formal concept c_{k+1} . Every concept c_{k+1} is a subconcept of c_k , or equal to c_k in the case the DT concept a is in the intent of c_k . Moreover, every immediate subconcept of c_k is among those concepts c_{k+1} , i.e., can be reached in one step from c_k . Therefore, every subconcept of c_k can be reached in a finite number of steps. Similarly, it can be demonstrated that the removal of DT concepts in the query corresponds to navigation links towards superconcepts.

Although DTs are able to eventually reconstruct the whole concept lattice through an exhaustive navigation, each zoom step has a very low complexity. If the deep extension of each DT concept is explicitly stored, each intersection can be computed in a linear time, either by using ordered inverted lists or bitmaps for storage.²² Since the number of intersections to be performed is no larger than the total number of DT concepts, the overall time complexity is $O(mn)$, where m is the number of DT concepts/attributes and n is the number of objects. The lazy computation paradigm used by DT, where relationships are inferred on demand, makes DT scalable to large and very large databases.²³ Most importantly, DT being entirely based on dynamic computations, it can easily deal with dynamic databases in which new objects are inserted, deleted or modified. The concept lattice, by converse, is a static, precomputed structure that cannot easily accommodate variations in the database, except for incremental object insertion [196]. For this reason, FCA techniques seem more suitable for data analysis of static sets rather than for intelligent information access in a dynamic environment.

²²See Sect. 8.1.

²³See Chap. 8.

The dynamic computation of dynamic taxonomies over arbitrary subsets of objects allows to considerably extend the expressiveness of the model. Whereas the concept lattice is derived by the AND-ing of attributes only, the combination of DT concepts using all the logic operations: AND, OR, and NOT is allowed in DT.

The informativeness of dynamic taxonomies is intermediate between the global and local views on concept lattices. We have shown above that every immediate sub-concept and superconcept of the current formal concept can be reached by selecting some DT concept in the dynamic taxonomy. Also, every attribute in the intent of the current formal concept appears in the dynamic taxonomy, as a DT concept whose count is equal to the size of the extent, i.e., a DT concept/attribute that is shared by all objects in the current focus/concept. Therefore, the local view on the concept lattice, which is used by several FCA tools, is fully included in DT. However, a dynamic taxonomy shows only extensional relation between the focus and each DT concept, whereas the concept lattice shows extensional relations between the focus and every combination of DT concepts, i.e., between the current formal concept and every other concepts. On one hand, FCA gives a complete picture of the conceptual structures laying in data, which is suitable to data mining tasks. On the other hand, DT is less informative, but it ensures the results displayed to users is manageable because they are expressed in terms of the original taxonomy. This reduces the cognitive effort of users and is a simpler and more intuitive representation than the line diagrams generally used in FCA.

3.4.3 Conclusion

In summary, we have shown that FCA and DT share a similar data model, and a similar navigation space that enables to tightly combine querying and navigation in a same search. The concept lattice has a strong value for data-mining tasks, but does not scale well, both computationally and visually, for information access. However, the concept lattice is still useful in information access as a characterization of the navigation space, where each concept corresponds to a focus. Compared to the concept lattice, DT is a dynamic view on it, in the form of a taxonomy, that scales up to large and very large infobases, and requires less cognitive effort. DT can be seen as an extension of local views on concept lattices, which consists in a concept and its neighbors. Because of the similarity of their data model, it should be possible to apply similar extensions to FCA and DT, as this was done for logics [102, 106]. Also, DT could be used as a front-end for the exploration of large concept lattices, and the ample body of results around FCA could be used to improve it, or even extend it towards data-mining tasks.

3.5 Semantic Web

The Semantic Web (SW) [23] is an evolving extension of the World Wide Web in which Web content can be expressed not only in natural language but also in a format

that can be read and used by software agents, thus permitting them to find, share and integrate information more easily. It comprises a variety of formally specified languages (RDF/S,²⁴ OWL²⁵), associated formats (e.g. RDF/XML, N3, Turtle, N-Triples) and related technologies for their management. In contrast to XML, which also allows expressing structured documents, SW languages have a clear semantic interpretation.

Resources (e.g. Web pages, books, people, places, hotels) can be described using RDF statements. RDF statements are actually object-attribute-value triples, e.g. `Yannis bornIn Athens`. Such descriptions may refer to *ontologies*, i.e. they can be considered as instantiations of ontologies. This allows expressing statements of the form `Yannis typeOf Professor` (i.e. Yannis is an instance of the class Professor) where the class Professor may be defined in an ontology that contains also the statement `Professor subclassOf Person`. This allows inferring that Yannis is a person although that statement has not been specified explicitly.

Regarding languages and formats:

- *XML* is a meta-language for defining markup. It provides a syntax for structured documents but does not impose any semantic constraint on the meaning of these documents.
- *XML Schema* is a language for restricting the structure of XML documents.
- *RDF* is a data model for objects (resources) and relations between them. It is actually a data model based on object-attribute-value triples. Such triples can be represented and exchanged in various formats including XML (i.e. RDF/XML) and TriG [43].
- *RDF Schema* is a vocabulary description language for describing properties and classes of RDF resources equipped with subsumption semantics. It allows expressing ontologies.
- *OWL* is a richer vocabulary than RDF Schema that allows expressing disjointness and cardinality constraints as well as other properties like equivalence, symmetry, transitivity.

In one sentence, an ontology is an explicit and formal specialization of a conceptualization [126]. Ontologies allow capturing domain knowledge using the classical knowledge representation mechanisms (classes, attributes, generalization/specialization hierarchies). Specifically, they comprise a finite list of terms and relationships between these terms. The terms denote concepts (classes of objects) of the domain, while the relationships allows forming class hierarchies (e.g. `Professor subclassOf Person`) and defining properties (e.g. `Yannis teaches InformationRetrieval`), as well as various value restrictions (e.g. only professors can teach courses). Ontologies can be expressed in RDF Schema and OWL (also see Sect. 8.4). The following section explains how OWL is based on a logical formalism, description logics, and how it can be compared to taxonomies.

²⁴<http://www.w3.org/TR/rdf-schema/>.

²⁵<http://www.w3.org/TR/owl-features/>.

3.5.1 Description Logics and OWL Ontologies

The purpose of OWL is to define ontologies for modeling and reasoning about application domains. OWL is mapped onto the description logics. Description logics provide the required theoretical background, such as proving the consistency of an ontology, or classifying concepts. The OWL standard includes three expressivity levels: OWL Lite, OWL DL and OWL Full. OWL Lite does not contain the notion of cardinality on roles, but inference is decidable and efficient. OWL Full uses all the primitives of OWL, and is the most expressive language (and fully includes RDF), but becomes undecidable. Here, we focus on OWL DL, because it is decidable, a good compromise between expressivity and inference efficiency, and well supported by most existing tools (unlike OWL Full). In the following, we recall the basic definitions of OWL DL, and we adopt the terms and notations of description logics rather than those of the OWL standard, because they provide a much more compact syntax, and because our focus is more on theory than on technology.

Description Logics (DL) are logical formalisms that have been adopted to represent and reason about ontologies [143], which are domain-specific knowledge bases. They have points in common with dynamic taxonomies such as concepts, subsumption between concepts, and instance relation between objects and concepts. There are however several important differences. First, concepts are not only concept names, but complex combinations of concepts by logical constructors. Second, the subsumption relations are derived automatically from a set of axioms modeling the domain. Third, relations (called *roles*) can be set between objects, so that the concepts to which an object belongs depend on other objects to which it is connected.

The definition of description logics is based on three kinds of entities: concepts, roles and objects. Given a domain of individuals (e.g., persons in genealogy, documents in bibliography), the concepts are sets of individuals that share some characteristics (e.g. *Women*); the roles are binary relations between individuals (e.g. *sibling*); the objects are particular individuals (e.g. BOB). The syntax gathers these three kinds of entities, plus constructors for building complex concepts and roles, which determine the expressivity of the description logic.

Definition 3.1 (Signature) The language of concepts of a description logic is characterized by a signature $S = (O, C_a, R_a, Cstr)$, where:

- O is a set of object names;
- C_a is a set of concept names (denoted c_i);
- R_a is a set of role names (denoted r_i);
- $Cstr$ is a set of constructors used to create complex concepts (denoted C_i) and complex relations (denoted R_i). Classes of constructors of the common description logics are represented by letters.

Table 3.4 Set of constructors of the description logic *SHOIQ*, with their syntax and interpretation

Letter	Definition	Syntax	Interpretation
S	top (most general concept)	\top	Δ_I
	bottom (most specific concept)	\perp	\emptyset
	conjunction of two concepts	$C_1 \sqcap C_2$	$C_1^I \cap C_2^I$
	disjunction of two concepts	$C_1 \sqcup C_2$	$C_1^I \cup C_2^I$
	concept negation	$\neg C$	$\Delta_I \setminus C^I$
	qualified universal quantification	$\forall r.C$	$\{i \in \Delta_I \mid \forall j : (i, j) \in r^I \rightarrow j \in C^I\}$
	qualified existential quantification	$\exists r.C$	$\{i \in \Delta_I \mid \exists j : (i, j) \in r^I \wedge j \in C^I\}$
O	at least one of objects	$\{o_1, \dots, o_n\}$	$\{o_1^I, \dots, o_n^I\}$
Q	minimal cardinality	$\geq n \ r.C$	$\{i \in \Delta_I \mid \#\{j \in C^I \mid (i, j) \in r^I\} \geq n\}$
	maximal cardinality	$\leq n \ r.C$	$\{i \in \Delta_I \mid \#\{j \in C^I \mid (i, j) \in r^I\} \leq n\}$
	exact cardinality	$= n \ r.C$	$\{i \in \Delta_I \mid \#\{j \in C^I \mid (i, j) \in r^I\} = n\}$

In OWL, objects are called *individuals*, concepts are called *classes*, and roles are called *properties*. The description logic OWL DL is based on the description logic *SHOIQ*, whose constructors are listed in Table 3.4 (the notation $\#S$ represents the cardinality of a set S). Each letter in *SHOIQ* corresponds to a set of constructors or to axioms (letter H and I , see below). This list of constructors determines the expressivity of OWL DL.

The second notion to be defined for a description logic is semantics. The main difference between classical logic and description logics is the interpretation function. Where classical logic interprets each formula as a truth value, description logics interpret each concept as a set of individuals, its instances, and each role as a binary relation between individuals. The interpretation function for the constructors of the description logic *SHOIQ* is given in Table 3.4.

Definition 3.2 (Interpretation) Let $S = (O, C_a, R_a, Cstr)$ be a signature. An interpretation $I = (\Delta_I, \cdot^I)$ of S is a set of individuals Δ_I , called the interpretation domain, and an interpretation function \cdot^I , that associates:

- to each object name o_i an individual $o_i^I \in \Delta_I$;
- to each concept name c_i a set of individuals $c_i^I \subseteq \Delta_I$;
- to each role name r_i a binary relation between individuals $r_i^I \subseteq \Delta_I \times \Delta_I$.

The third notion to be defined for a description logic is the knowledge base. In OWL, a knowledge base is called an *ontology*. A knowledge base divides the knowledge in two parts: the *terminological* part, which represents general knowledge, true for all individuals, and the *assertional* part, which represents particular knowledge, only true for particular individuals.

Definition 3.3 (Knowledge base) We define a knowledge base as a pair $\Sigma = (T, A)$, where:

- T is the terminological part (T-Box), represented by a set of axioms like “ C_i is subsumed by C_j ”, denoted $C_i \sqsubseteq C_j$, which means that every instance of C_i is necessarily an instance of C_j . The equivalence of two concepts, denoted by $C_i \equiv C_j$, is equivalent to the two axioms $C_i \sqsubseteq C_j$ and $C_j \sqsubseteq C_i$;
- A is the assertional part (A-Box), represented by a set of assertions like $o : C$ and $(o_i, o_j) : R$, which respectively means that “ o belongs to the concept C ”, and “a connection R exists between o_i and o_j ”.

Moreover, the description logic *SHOIQ* allows additional axioms in the knowledge base (letters H and I). The following axioms on roles can be added to the T-Box:

- $r_i \sqsubseteq r_j$ (called role hierarchy);
- r_i inverse of r_j ;
- r functional;
- r transitive.

In the following, the notation r^{-1} is used to denote the role that is defined to be the inverse of the role r , when defined.

Figure 3.8 presents the T-Box of an example knowledge base Σ_{ex} , with 3 concepts (*Person*, *Team*, *Bigteam*), and 2 roles (*hasleader*, *hasmember*). An instance of *Team* is defined as having a leader and at least 2 members. An instance of *Bigteam* is defined as having a leader and at least 3 members. The 3 axioms on roles say that every leader is a member, *isleaderof* is the inverse of *hasleader*, and there is a unique leader per team. To help the reading of knowledge bases, concepts are written with an uppercase, relations in lowercase, and objects in uppercase letters. Figure 3.9 presents the A-Box of Σ_{ex} , with the description of the team LIS, leded by OLIVIER and composed of 2 other members, PIERRE and SEBASTIEN.

<i>Person</i>	\sqsubseteq	\top
<i>Team</i>	\equiv	<i>hasleader</i> . <i>Person</i>
		$\sqcap \geq 2$ <i>hasmember</i> . <i>Person</i>
<i>Bigteam</i>	\equiv	<i>hasleader</i> . <i>Person</i>
		$\sqcap \geq 3$ <i>hasmember</i> . <i>Person</i>
<i>hasleader</i>	\sqsubseteq	<i>hasmember</i>
<i>isleaderof</i>	\sqsubseteq	inverse of <i>hasleader</i>
<i>hasleader</i>	\sqsubseteq	functional

Fig. 3.8 Example of T-Box

OLIVIER	:	<i>Person</i>
SEBASTIEN	:	<i>Person</i>
PIERRE	:	<i>Person</i>
LIS	:	<i>Team</i>
(LIS, OLIVIER)	:	<i>hasleader</i>
(LIS, SEBASTIEN)	:	<i>hasmember</i>
(LIS, PIERRE)	:	<i>hasmember</i>

Fig. 3.9 Example of A-Box

Given a knowledge base Σ , some interpretations are distinguished as *models* of Σ .

Definition 3.4 (Model) Let $\Sigma = (T, A)$ be a knowledge base and I an interpretation, for the same signature S . We call I a model of Σ , denoted $I \models \Sigma$, when:

- for each $C_i \sqsubseteq C_j$ in T , $C_i^I \subseteq C_j^I$;
- for each $o : C$ in A , $o^I \in C^I$;
- for each $(o_i, o_j) : R$ in A , $(o_i^I, o_j^I) \in R^I$.

The definition of *SHOIQ* gives additional properties to be checked:

- for each $r_i \sqsubseteq r_j$ in T , $r_i^I \subseteq r_j^I$;
- for each r_i *inverse* of r_j in T ,
 $r_i^I = \{(i, j) \in \Delta_I \times \Delta_I \mid (j, i) \in r_j^I\}$;
- for each r *functional* in T ,
 $(i, j) \in r^I \wedge (i, j') \in r^I \rightarrow j = j'$;
- for each r *transitive* in T ,
 $(i, j) \in r^I \wedge (j, k) \in r^I \rightarrow (i, k) \in r^I$.

From this notion of model, statements and their inference from a knowledge base can be defined: classification and instantiation. The *classification* statements establish subsumption relations between concepts, and hence help to organize or classify them. The *instantiation* statements establish the belonging of objects to concepts, and hence help to place objects in the concept classification.

Definition 3.5 (Classification and Instantiation) Let Σ be a knowledge base. Two kinds of statements can be inferred from Σ :

- Classification: $\Sigma \models C_i \sqsubseteq C_j$ iff $I \models C_i \sqsubseteq C_j$, for every model I of Σ ;
- Instantiation: $\Sigma \models o : C$ iff $I \models o : C$, for every model I of Σ .

For instance, it can be proved in our example that every big team is also a team: $\Sigma_{\text{ex}} \models \text{Bigteam} \sqsubseteq \text{Team}$. Also, OLIVIER being the leader of LIS, hence a member of it, and SEBASTIEN and PIERRE being members too, we can prove that LIS is a big team, i.e. $\Sigma_{\text{ex}} \models \text{LIS} : \text{Bigteam}$.

Clearly, Semantic Web languages allow defining complex semantic structures that are more expressive than taxonomies and taxonomy-based classifications. The availability of such structures allows formulating and answering queries that could not be expressed precisely using keyword queries, e.g. *find all used cars manufactured in Germany that were owned by a citizen of Greece*. However the formulation of such queries is a rather difficult and hard task for end users: the above query can be translated by the concept

$$\text{Car} \sqcap \exists \text{user} . \top \sqcap \exists \text{manufacturer} . \{GERMANY\} \sqcap \exists \text{owner} . \exists \text{citizen}^{-1} . \{GREECE\}.$$

Therefore it is worth investigating methods that allows creating complex query patterns in an intuitive and gradual/progressive manner, and exploratory search is certainly an approach that is worth investigating.

Therefore, an interesting question is how an OWL ontology, i.e. a DL knowledge base, could fit into the framework of dynamic taxonomies. The key notions are the classification and instantiation statements that can be inferred from a knowledge base.

- Classification statements define a subsumption ordering over concepts, which can play the role of a taxonomy. This forms a multi-dimensional taxonomy because an individual can belong to an arbitrary number of concepts, and concepts can have several super-concepts.
- Instantiation statements determine whether an object is an instance of a concept, and hence define the deep extension of concepts.

Therefore, all ingredients are present to apply the framework of dynamic taxonomies onto Web ontologies. The important differences with the usual application of DTs is that (1) the concepts may be complex and in infinite number, (2) the taxonomy shape and extensions are defined through logical inference, and (3) there are relations between objects that may be followed to reach a set of related objects from the current focus (e.g., *the Greek owners of some set of cars*).

3.5.2 Semantic Web and Exploratory Search

The question is how dynamic taxonomies and faceted exploration could be exploited for ontologies and ontology-based descriptions. We could identify the following challenges:

- Ability to navigate through properties of resources of any type and to make selections based on properties of other, semantically related, types. Note that most facet browsers provide an interface to a single type of resource. Including multiple types, could lead to an explosion of the number of properties and thus the number of available facets.
- Ability to handle any RDF/S or OWL dataset without any additional configuration.
- Efficient computation of zoom points.

Below we discuss in brief some possible approaches that can be followed. Let us first consider the case where resources are classified with respect to ontologies that have a plain taxonomic structure, i.e. comprise classes organized hierarchically through the `subclassOf` relation and these classes have no properties. This case matches exactly with the data model of taxonomy-based sources therefore the provision of dynamic taxonomies and faceted search is straightforward. If all SW data are loaded in main memory then the implementation of exploratory search can be based on the existing SW main memory APIs. If on the other hand ontologies and ontology-based descriptions are stored in secondary memory, specifically in a triplestore that offers a query language (e.g. RQL [157], or SPARQL [8]), then that language could be used for computing the zoom points.

Let us now consider the general case, i.e. the case where ontologies also contain properties (whose domain and range are classes or literal types). The question is how properties should be exploited during browsing. Below we sketch some approaches that could be adopted:

- *View-based* approach

If the ontologies and the ontology-based descriptions of the resources are stored in a triple-store that offers a query language, then that language could be used for defining the desired view(s) for exploration. Specifically, one may define several views each having the form of a taxonomy-based source.

If some attributes have a big number of distinct values, then methods that group these values and define taxonomies on the fly (like in [61]) can be adopted in order to avoid overloading the user.

- *Generic* approach

Another approach would be to directly apply it on RDF or OWL, without the need of a special configuration. Examples of systems that attempt this include: /facet [139], Ontogator [175], BrowseRDF [200] and are described in more details below.

Below we describe in brief some view-based and generic approaches that have been proposed or used.

- Ontogator [175]

It is an open source tool that has been applied in several semantic portals on the Web. It supports several views and their configuration is done using RDF. It employs (Prolog) rules for articulating values with background knowledge and follows the view-based approach. Labeling schemes (e.g. [17]) are employed for speeding up the computation of zoom-in points.

- BrowseRDF [200]

It is an RDF browser for semi-structured RDF data. The mapping of RDF to Dynamic Taxonomies (DT) is the following:

	DT	↔	RDF
	object	↔	RDF subjects
	facet	↔	RDF predicates
	restriction-values/zoom-in points	↔	RDF objects

During interaction, the user not only is able to restrict the focus by selecting a value, but he can also exploit the following operators:

- *Existential* selection

For example, consider a knowledge base that describes persons and their relationships. Existential selection allows selecting all resources (e.g. `persons`) with a `spouse`, or all resources without a `spouse`. This means that the user does not have to select a particular value.

- *Join* selection

This operator allows forming restrictions on paths of properties, for example, all resources who know somebody, who in turn knows somebody named `Tonia`.

– *Inverse selection*

This operator allows forming conditions on properties that point to (instead of originating from) the current objects.

As the facets could be many, metrics for automatically ranking facets according to their quality (for browsing) are introduced.

• */facet* [139]

It is a facet browser for Semantic Web repositories and it has been applied for browsing and searching museum collections. The repository may contain more than one RDF schemas and thesauri. It supports several resource types (i.e. `rdf:type` is treated as a facet) and during browsing the user can switch the type of sought objects. The user, apart from browsing, is able to search a facet, and autocompletion facilities (during typing) are also provided. Regarding the presentation of the facets, some visualization and interaction plug-ins are supported. For instance, geographical data are displayed in an interactive map, while time-related facets (e.g. `dc:date`) are visualized using a timeline plug-in. However, runtime reasoning causes delays and for this reason some deductively derived data are materialized.

Examples of other systems that follow similar approaches include: MuseumFinland [147], and Longwell RDF Browser [7].

In Sect. 5.5, we present an extension of DTs that follows the generic approach. It applies to any OWL DL ontology, and is directly defined on top of OWL. It allows users to gradually build complex queries by successive selections in dynamic taxonomies, which are computed with the help of existing OWL reasoners. It handles different types of resources, and provides most OWL operators (existential selection, inverse property, join selection) through both querying and navigation.

3.5.3 Conclusions

In conclusion: (a) SW languages and formats can be used for representing and exchanging taxonomies and taxonomy based descriptions, (b) SW technologies (e.g. triple stores, query languages) can be used for storing and querying taxonomy-based sources, (c) in the SW user access is usually done through specialized and proprietary software agents. The application of the interaction paradigm of dynamic taxonomies and faceted search requires tackling issues similar in spirit to those for supporting exploratory search over relational databases (for more see Sects. 7.1 and 8.2).

In addition, and as pointed out in [292], we would like to remark that in a very broad domain such as the set of all Web pages, it is not easy to identify the classes of the domain because the domain is too wide and different users, or application needs, conceptualize it differently, e.g. one class of the conceptual model according to one user may correspond to a value of an attribute of a class of the conceptual model according to another user. For example, Fig. 3.10 shows two different ways to

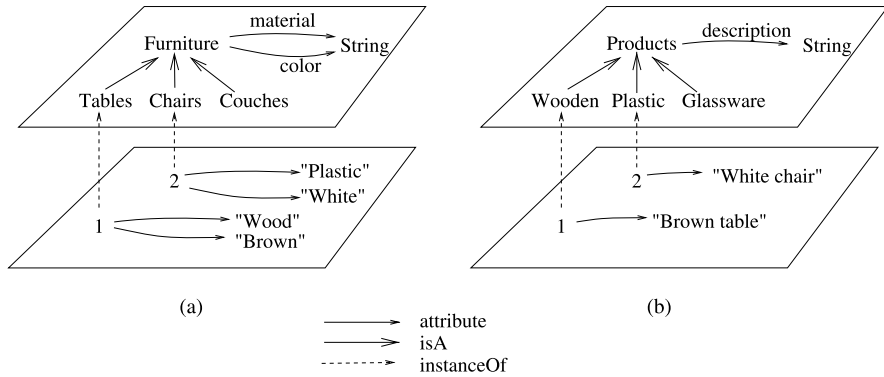


Fig. 3.10 Two different ontologies (conceptual models) for the same domain

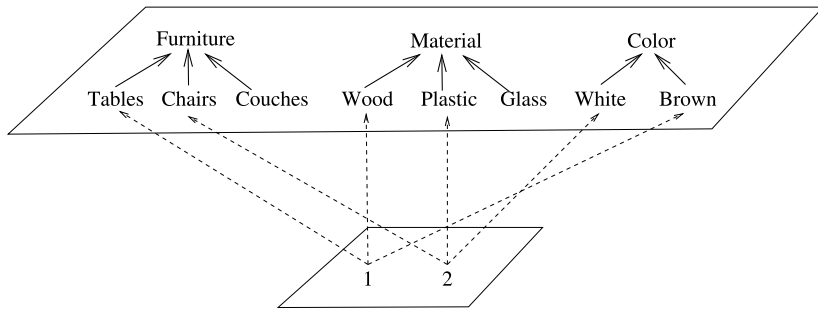


Fig. 3.11 An ontology that consists of terms and subsumption links only

conceptualize the same domain. Our example shows only two objects of the domain which are denoted by the natural numbers 1 and 2.

The conceptual model of Fig. 3.10(a) seems appropriate for building an information system for a store that sells furniture, while the conceptual model of Fig. 3.10(b) seems appropriate for building an information system for a supermarket.

We can say that the classes of the ontology (a), i.e. the classes Tables, Chairs and Couches, have been defined in order to distinguish the objects of the domain according to their *use*. On the other hand, we can say that the classes of the ontology (b), i.e. the classes Wooden, Plastic and Glassware, have been defined in order to distinguish the objects of the domain according to their *material*. This kind of distinction is useful for a supermarket, as it determines (in a degree) the placement of the objects in the various departments of the supermarket.

Figure 3.11 shows an ontology for the same domain which consists of terms and subsumption links only. This ontology seems to be more application independent. All criteria (characteristics) for distinguishing the objects are represented in the same way, in a truly faceted approach.

This modeling approach is simple and scalable, and remains to be effective as long as the access/exploration criteria for the objects are based on their properties and attribute values, and not on their associations with the rest objects. To access and explore objects based on their associations with other objects, techniques like those described in Sects. 5.5 and 7.1 can be applied.

Chapter 4

User Interface Design

Moritz Stefaner, Sébastien Ferré,
Saverio Perugini, Jonathan Koren, and Yi Zhang

*“Design is not just what it looks like and feels like.
Design is how it works.”*

Steve Jobs, 1955–

As detailed in Chap. 1, system implementations for dynamic taxonomies and faceted search allow a wide range of query possibilities on the data. Only when these are made accessible by appropriate user interfaces, the resulting applications can support a variety of search, browsing and analysis tasks.

User interface design in this area is confronted with specific challenges. This chapter presents an overview of both established and novel principles and solutions. Based on a definition of core principles (see Sect. 4.1) and challenges (see Sect. 4.2), we define a taxonomy of navigation modes observed in existing applications (see Sect. 4.3). On that basis, design patterns for enabling these navigation modes in user interfaces (see Sect. 4.4) as well as extensions and related approaches (see Sect. 4.5) are discussed. The chapter closes with an approach to personalizing faceted search (see Sect. 4.6).

4.1 Principles

Extending traditional models of Information Retrieval, *search for digital resources* has lately been widely recognized as multi-step processes [32, 130, 179, 228]. To follow the terminology introduced in [130], a search usually involves an *initial constraint definition*, followed by an *orienting and refinement* phase based on first inspections of the result, and finished with a closer examination of individual results in the so-called *endgame*.

In this context, the exploration of dynamic taxonomies [236] with *facet browsers* is often seen as a most promising candidates for “rich exploration of a domain across a variety of sources from a user-determined perspective” [155]. These make different aspects of the underlying data accessible in parallel. Selecting one of the values, and thus filtering the result set, restricts the available metadata values only to those occurring in the results. Consequently, the user is visually guided through an iterative process of query refinement and expansion, never encountering situations with zero results.

Advanced Search

Type a query into the search box or use the fields below to construct your query

Include **ALL** of the following:

Keyword Anywhere

Add Another

But does not include:

Keyword Anywhere

Add Another

Limit my search results by:

Branch: Any

Available now:

Language: Any Language

Audience: Any

Format: Any Format

Date Published: Between YYYY and YYYY

Search Reset

Fig. 4.1 The advanced search interface for the Oakville Public library at <http://opl.bibliocommons.com/search>

Applications for faceted search and dynamic taxonomies offer the following key features to support a wide range of search and browsing tasks:

- **Unrestricted query formulation over multi-dimensional classification**

Facet browsing applications impose no restrictions, in which order, or in which granularity filters are applied on a result set. Filters stem from various, orthogonal dimensions that can be combined by Boolean operators. This allows the formulation of complex queries, such as “All documents created before date A, related to topic B, and of file type C or D”. The equal treatment of multiple dimensions differs from, e.g. typical web site structures or file systems, where a single taxonomy is the pre-dominant organization principle, and other metadata are only supplements for sorting or filtering.

- **Poka yoke: no more empty result sets**

One of the core principles of dynamic taxonomies is to restrict the available filtering options in the given focus to only those, which will lead to a non-empty result set. Hence, the user can never run into a situation with zero results. This is opposed to the process in a typical *advanced search* situation, where first a complex boolean query is constructed, which is then evaluated on demand (see e.g. Fig. 4.1). That, however, can result in empty result sets, often without further indication on which part of the query could be relaxed in order to retrieve some results. The exclusion of potentially frustrating situations by design is often referred to as poka-yoke principle.¹

¹See e.g. <http://en.wikipedia.org/wiki/Poka-yoke>.

- **Orienteering and domain understanding**

It is a common pattern to visualize the number of occurrences of a concept in the given focus. The simplest option is to provide it after the concept label (e.g. “Europe (5)”). Advanced techniques include the application of visual indicators, such as bar height or small bar charts (see Sect. 4.4.7).

This provides valuable *information scent* [212], i.e. “a user’s (imperfect) perception of the value, cost, or access path of information sources obtained from proximal cues” [317]). Orienteering, or “directed situated navigation” [284] is the process of reaching a goal through a series of small actions, supported by continuous evaluation of the respective focus. In this context, knowing beforehand how many resources to expect after adding a concept as a filter, can be a valuable indicator of the utility of the filtering action. Additionally, this principle can be extended in order to foster domain understanding by learning about characteristic metadata distributions (see Sect. 4.4.7).

4.2 Challenges

The prototypical facet browsing application has at least two main interface areas: one for presenting facets and their values, one for displaying the result set. Additional components might include a detail view for selected resources and a breadcrumb strip for filter summary and selection history navigation (see Sect. 4.4.5).

Based on this basic setup, a number of dimensions can vary in the system and user interface design, and need to be carefully decided upon:

- Which is the data type of the different facets—nominal, hierarchical, ordinal, real valued?
- How are facet values presented to the user? Are all facets and values visible, or only a selection?
- Can the user select multiple values per facet? If so, does this result in conjunctive or disjunctive queries?

Based on these fundamental considerations in setting up a faceted navigation scheme, and designing an appropriate interface, the following recurring challenges in designing these systems will have to be tackled [131, 133, 163]:

- **Boolean query logic** A selection of single concepts from different facets is usually understood as conjunction (AND-query). If, however, multiple values within one facet are selectable (for instance, “red” and “green” from the “color” facet), depending on context and data set, either a conjunctive (“red” AND “green”) or a disjunctive (“red” OR “green”) interpretation are conceivable. If an application only uses one of these selection modes, this needs to be communicated to the user; if both are possible, separate controls for both modes will be needed (see Sect. 4.4.1).

- **Cluttered interfaces** The paradigm of making all filter options available in parallel naturally leads to the challenge of having to fit many controls and text fields on the user screen. Hence, clear visual structure and hierarchy as well as strategies to reduce visual clutter are vital. If a full exposure of all facets is not possible due to size constraints, strategies and user controls for showing and hiding, or expanding and collapsing facets will have to be integrated (see Sects. 4.4.2 and 4.4.3).
- **Incorporating keyword search** A free-form keyword field in order to search for arbitrary terms in addition to the pre-defined classification scheme is a “key component to successful faceted search interfaces” [131]. One source of confusion can be the question, if the search field will act as a plain text filter (e.g. searching over titles and descriptions of the resources) or if it will also match classification terms. A third conceivable option is a “search within the results”, which just filters the result display, but does not act as a full-fledged facet. In either case, the relation of the free-form search to the rest of the filters has to be signaled clearly in order to avoid misconceptions (see Sect. 4.4.4).
- **Change blindness** Change blindness is a well-known psychological phenomenon [222]: a person viewing a visual scene apparently fails to detect large changes in the scene, if the change in the scene coincides with some visual disruption such as a saccade (eye movement) or a brief obscuration of the observed scene or image. This situation often occurs in web applications, where the web page briefly flashes after actions demanding a new server request. In this context, animated transitions can facilitate perception of changes in user interface design [135, 286, p. 84]. Perception of change is especially important for facet browsing, as the sudden disappearance of list items after a click can be a source for misconceptions and confusion. Besides animation, clear marking of the current focus and the resulting effects are recommended (see Sect. 4.4.6).

4.3 Navigation Modes

As a basis for comparing user interface design patterns in the next section, this section defines and illustrates different navigation modes, that enable the user to navigate the available information space by consecutively applying operators on the query.

Given an infobase over a taxonomy (T, \leq) of concepts, a query is a Boolean combination of concepts. We recall the extension of such a query can be computed from the extensions of concepts by applying set operations: intersection for conjunction (*and*), union for disjunction (*or*), and complement for negation (*not*).

From this perspective, browsing an infobase consists in navigating from query to query. This is more general than defining browsing as navigating from sets of objects to sets of objects, because every query determines a set of objects, its extension, and not all navigation modes can be defined as a function from sets of objects to sets of objects. The queries are constructed by following navigation links or using interface controls. Most navigation links are provided by dynamic taxonomies, which also summarize the extension of the current query.

Based on an analysis of existing applications, we can distinguish the following navigation modes:

- *zoom-in* makes the query more specific,
- *zoom-out* makes it more general,
- *shift* replaces a part of the query by a related concept,
- *pivot* replaces the whole query by a related concept,
- *slice-and-dice* allows the disjunctive selection of multiple concepts within a facet,
- *range selection* offers the options to specify query intervals within ordinal or real value facets.

The change from a query to another query, and hence, from a focus to another focus, is defined as a *navigation link*. A navigation link is decomposed into a *selection* and a *navigation mode*. This means that a same selection can be used in different ways to reach different foci. In the simple case, a selection is a concept in the dynamic taxonomy of the current focus. In the general case, a selection is the disjunction of the concepts that are selected in the dynamic taxonomy (e.g., France or Germany or Italy). Controls in the interface can be activated to apply modifiers on such selections: adding negation (e.g., not (Animal or Plant) from the selection of Animal and Plant), replacing equalities by inequalities (e.g., date \geq 2002 from the selection of date = 2002). Given those selections, the above navigation modes can be reduced to only two primitive navigation modes, *zoom* and *pivot*:

- *zoom-in* is a zoom on a selection whose extension contains some objects of the focus, but not all,
- *zoom-out* is a zoom on a selection whose extension contains all objects of the focus,
- *shift* is a combination of zoom-in and zoom-out,
- *pivot* is a basic mode,
- *slice-and-dice* is a zoom on a selection with disjunction,
- *range selection* is a zoom on a selection with inequalities.

An additional navigation mode is *querying-by-examples*, which defines the query from the selection of a set of objects, the examples.

The definitions of navigation modes rely on the fact that queries can be put in conjunctive normal form, i.e. conjunctive sets of simpler queries. For instance, France and not date \leq 2000 and (Building or Landscape) is equivalent to {France, not date \leq 2000, Building or Landscape}.

In the following, these interaction modes are illustrated with an example scenario using Camelis,² a system for browsing a personal photo collection spanning the period 1999–2007. This collection contains 5,820 photos, which are described by date, location, event, type, visible persons and objects, and EXIF descriptors (e.g., time, flash, orientation).

²The version used here is 1.4, and can be downloaded at <http://www.irisa.fr/LIS/ferre/camelis/>.

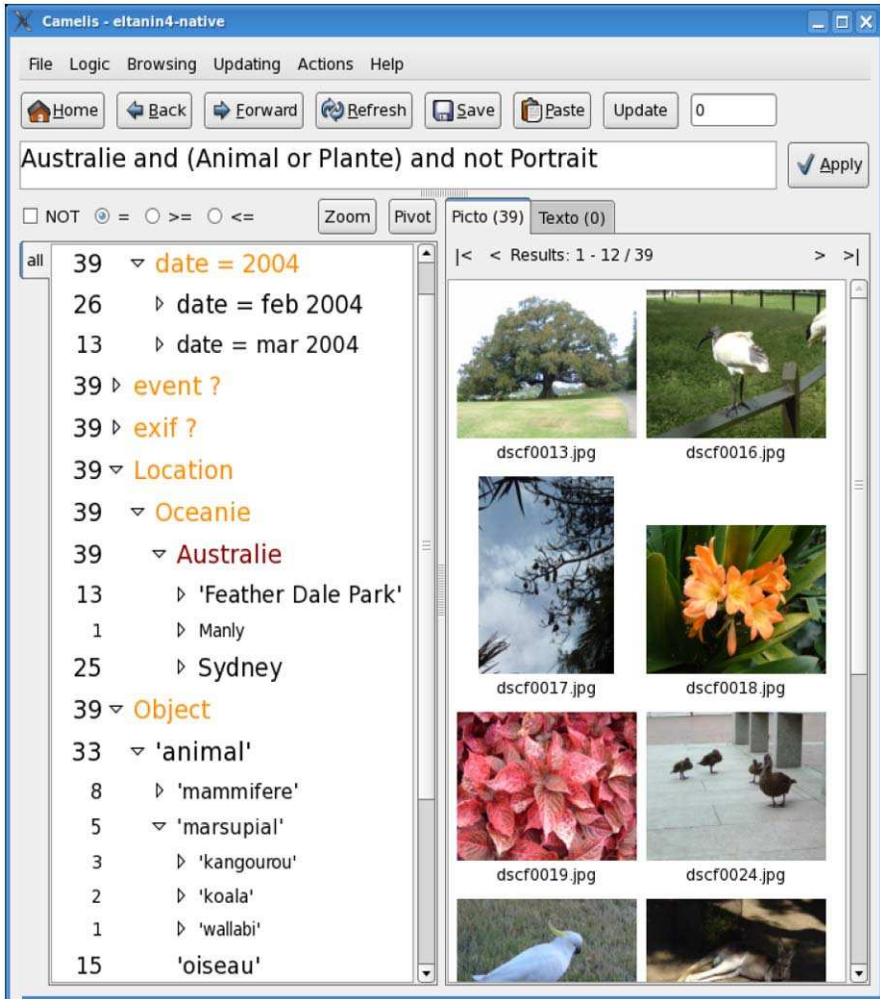


Fig. 4.2 The graphical interface of CAMELIS

Figure 4.2C shows a screenshot of Camelis. The current query is at the top. The extension of this query, i.e., the current focus, is at the right, where each object is represented by a thumbnail or a text snippet depending on its type. The dynamic taxonomy is at the left, in the form of concept trees whose nodes are expanded on demand. The number at the left of each concept represent its count in the current focus, and the font scale is logarithmically proportional to this count. At the top of the dynamic taxonomy, there are check and radio buttons to modify the current selection (insertion of negation and inequalities), and two buttons for applying the two primitive navigation modes (zoom and pivot).

4.3.1 Zoom-in

First, suppose some user, say Lisa, wants to find some photos from a trip in Australia at the conference ICFCA'04. She first expands the concept `Location`, and finds she has photos from Europe (5,346), Africa (162), and Australia (148). After selecting the concept `Australia`:³

- the query becomes `Australia`,
- the extension displays 12 photos (out of 148),
- the concept `Australia` has now maximal font scale because all photos in the extension belong to it, and it is automatically expanded to show sub-locations of Australia (Lisa finds that she has been mainly in Sydney (105), and in the Blue Mountains (18)),
- the concepts `Europe` and `Africa` are no longer visible, because they are no longer relevant, i.e., count = 0.

Now she expands the concept `Type` and sees there are different types of photos: buildings (29), animals (34), and plants (6). She becomes interested in Australian organisms, so she selects both `Animal` and `Plant`, which leads her to the refined query `Australia and (Animal or Plant)`, whose extension contains 40 photos. One of these photos is a portrait, which Lisa does not want, so she selects the negation of `Portrait`. This leads her to the new query `Australia and (Animal or Plant) and not Portrait` (39 photos). By expanding more concepts, she discovers that these photos were taken in February and March 2004, mainly in Sydney and at the Feather Dale Park, and that 5 photos of three different species of marsupials are present: kangaroo, koala, wallaby.

Figure 4.2C shows the interface obtained after the previous navigation operations. At this stage, Lisa can either browse the 39 photos in CAMELIS, or launch a slideshow in an external application.

These three navigation steps lead to local views with increasingly more precise queries, and hence increasingly smaller extensions. This is called *zoom-in* navigation, because it corresponds to moving towards smaller extensions. Its principle is to specialize the current query q by the selection x . A simple definition of the resulting query would be q and x , but this would entail redundancy in queries: e.g., `Australia and Sydney` which is equivalent to `Sydney` because `Sydney` is subsumed by `Australia` in the taxonomy. A better definition is to replace by x every part of the query that subsumes x :

$$q \rightsquigarrow (q \setminus \{y \in q \mid x \leq y\}) \cup \{x\} = \text{Min}_{\leq}(q \cup \{x\}).$$

The extension of the new query is $\text{extension}(q) \cap \text{extension}(x)$. Therefore, in case the extension of x contains the extension of the query, the new query has the same

³There are French words in the screenshot as it is a personal photo collection, but English translations are used in the text for better consistency.

extension as the query q . So, we restrict zoom-in to selections x such that

$$\text{extension}(q) \cap \text{extension}(x) \neq \text{extension}(q),$$

i.e., to selections whose extension contains some of the objects of the focus, but not all.

4.3.2 Zoom-out

During navigation, the user may want to remove or generalize concepts in the query so as to reach larger extensions: this is the *zoom-out* navigation mode. For instance, Lisa realizes she needs more photos of animals and plants. The back button can be used to retract the previous refinement. Hence if she wants to remove the first refinement *Australia*, she needs to move three steps backwards, and then re-select the last two refinements. She could also edit the query by hand, but users usually prefer to navigate rather than to edit queries [121].

Besides, selections whose extension contains all objects in the focus, i.e. $\text{extension}(q) \subseteq \text{extension}(x)$, cannot be used for zoom-in. This makes them available for zoom-out. When such a selection is part of the query, it is removed from the query:

$$q \rightsquigarrow q \setminus \{x\}.$$

For instance, if Lisa selects *Australia*, the new query is (*Animal or Plant*) and not *Portrait* (282 photos from many locations). When such a selection subsumes some query parts, it replaces such parts in the query:

$$q \rightsquigarrow (q \setminus \{y \in q \mid y \leq x\}) \cup \{x\} = \text{Max}_{\leq}(q \cup \{x\}).$$

For instance, if she selects *Pacific*, the new query is (*Animal or Plant*) and not *Portrait* and *Pacific*. It is now clear how zoom-in and zoom-out can be reduced to a single primitive navigation mode. When zooming on a selection, the relationship between this selection and the current query determines whether this is a zoom-in or a zoom-out.

Compared to existing approaches, i.e., a list of removable concepts, this approach has three advantages: (1) it is integrated into the dynamic taxonomy, (2) it allows the replacement of a concept by a more general one, and (3) it extends to selections with disjunction and negation by extending the ordering \leq to such selections.

4.3.3 Shift

Zoom-in and zoom-out can be combined in two forms of *shift* navigation modes. From the previous query *Australia* and (*Animal or Plant*) and not

Portrait, Lisa first chooses to zoom-in on the concept `Plant`, resulting in the query `Australia` and not `Portrait` and `Plant` (6 photos). This is her starting point for shift navigation.

At this point, Lisa sees that 1 photo has also the type `Landscape`, which interests her. She selects this concept (zoom-in) and, since the result has only 1 photo, she generalizes it by removing the concept `Plant` from the query (zoom-out). Therefore, she has executed a shift from Australian plants (6 photos) to Australian landscapes (80 photos), replacing in the query the concept `Plant` by the concept `Landscape`. From there, she performs a new shift from the concept `Landscape` to the concept `Building`, resulting in 28 photos of Australian buildings. These navigation steps are suggested and supported by photos belonging to two concepts, i.e., by extensional relations [236]. This illustrates the relevance of assigning several types to photos, which is common in this photo infobase. The same would apply to persons visible on photos, as a photo can contain several people.

However, the same does not apply to locations, as a photo cannot be taken in two incomparable locations (e.g., in Australia and in Europe). Nonetheless, it is still possible to shift between locations through the taxonomy of locations. Suppose Lisa wants to find building photos from other locations. She first generalizes `Australia` by `Location` in the query (zoom-out), and then browses suggested locations before selecting `Spain` (zoom-in). Thus, she has performed a shift from Australian buildings to Spanish buildings, and find 48 photos (mainly churches taken in the north-west of Spain in 2003).

The former form of shift is a zoom-in/zoom-out combination, and can be qualified as *extensional* because it relies on extensional relations in the infobase. The latter form of shift is a zoom-out/zoom-in combination, and can be qualified as *conceptual* because it relies on conceptual relations in the taxonomy.

4.3.4 Pivot

The user may not remember a concept she wants to use to refine the query, but she can find it through another query. For instance, suppose Lisa wants to retrieve the photos of the building of some town. She does not remember which town it is, but she remembers that the ICFCA conference took place there in 2004. Therefore, she can first reach the query `event contains "ICFCA" and date = 2004` by zoom-in navigation. The resulting extension shows photos of ICFCA'04, and the dynamic taxonomy shows relevant information about these photos, such as precise dates, locations, and so on. By browsing the dynamic taxonomy, she discovers that Sydney, in Australia, is the location of ICFCA'04. Then, she can make the query become `Sydney`, and refine it to the desired query `Sydney` and `Building` by zoom-in. The concept `Sydney` plays the role of a *pivot* between the two queries.

Pivot navigation relies on the ability of DTs to answer queries not only by a set of objects (the extension), but also by a set of concepts (the dynamic taxonomy). In previous navigation modes, these concepts were added or removed from the query,

whereas here they are used as new queries. Given a query q and a selection x , the query transformation is defined by

$$q \rightsquigarrow x.$$

Therefore, pivot navigation is a way to restart a search from the results of a first search. This kind of navigation has already been applied in collaborative web-sites [189, 335].

There is an interesting analogy with natural language. Indeed, the query above can be rephrased as “photos of buildings in the town, *where* the ICFCA conference took place in 2004”. The idea of pivot is reflected by the fact that Sydney occurs in the main sentence as “town”, and in the relative sentence as the relative pronoun “where”. The relative pronoun indicates which facet to browse for a pivot: e.g., “where” indicates a location, “when” indicates a date, and “who” indicates a person. Iterated pivot navigation then corresponds to nested relative sentences, such as “photos of buildings in the town, *where* the ICFCA conference took place in the year, *when* I also visited Hinterzarten”. The first pivot to be applied is the year 2004, and the second pivot is the town Sydney.

4.3.5 *Slice and Dice*

Section 4.3.1 shows how the query (France or Italy) and Building can be reached by performing a zoom-in successively on France or Italy and Building, thus selecting French and Italian buildings. The disjunction is introduced because both concepts France and Italy were selected when the zoom mode was activated. Now suppose Lisa wants to extend the selection to landscapes, while retaining the current selection of locations. She just has to extend the selection Building to the selection Building or Landscape, and activate the zoom mode. Because the new selection is more general than the old one, the zoom is interpreted as a zoom-out. According to the definition of zoom-out (Sect. 4.3.2), the new query is (France or Italy) and (Building or Landscape). Now, Lisa wants to refine the selection to Italy only. To this end, Lisa unselects France in the selection of locations, and applies the zoom mode. Because the new selection is more specific than the old one, the zoom is here interpreted as a zoom-in. According to the definition of zoom-in (Sect. 4.3.1), the new query is Italy and (Building or Landscape). This short navigation scenario demonstrates that in a query each facet can be refined and extended independently from other facets, simply by applying the zoom mode on the new selections. In fact, it is not necessary that the different concepts in a selection belong to the same facet, while this is the most common case.

4.3.6 Range Selection

Range selection is similar to slice-and-dice (Sect. 4.3.5), except disjunctions of concepts are replaced by inequalities as selections. This makes sense because an inequality `date >= 2002` is equivalent to the infinite disjunction `date = 2002` or `date = 2003` or `...`. Then, every range is the composition of two inequalities. For instance, the date range `date in [2002, 2007]` is equivalent to `date >= 2002` and `date <= 2007`. Therefore every range can be reached by two successive zoom-in steps on inequalities: one for the lower bound, and the other for the upper bound. It is assumed that the user interface allows the selection of inequalities even if the dynamic taxonomies contains only values (e.g., `date = 2003`).

Starting from `date >= 2002` and `date <= 2007` and France, the date range can be refined by zooming on `date >= 2003` or `date <= 2006` (zoom-in), or extended by zooming on `date >= 2000` or `date <= 2008` (zoom-out). The upper bound of the range can also be removed altogether by zooming on `date <= 2007` (zoom-out); and similarly for the lower bound. The above formulas are used to give the logic of the navigation, and to reduce range selection to basic navigation modes; but a user interface may render it in a more graphical way, e.g., with the help of a double slider on a scale covering the relevant values.

4.3.7 Querying by Examples

A query can be determined by the selection of a subset of objects, thus supporting querying by examples. The idea is to construct the query as a conjunction of all most specific concepts which are shared by the selected objects O :

$$q \rightsquigarrow \text{Min}_{\leq} \{y \in T \mid O \subseteq \text{extension}(y)\}.$$

For instance, suppose Lisa starts with Australia and not Portrait. While browsing photos in the result, she sees interesting photos of buildings (e.g., 2 photos of the Opera, and 1 photo of the Harbour Bridge), and she would like to find more. By selecting them she moves to a new query that is the conjunction of the concepts shared by those 3 photos. As usual with this form of navigation, the resulting query is very specific and she receives no additional photos. At this stage, Lisa can use zoom-out navigation to generalize the query. Unlike approaches based on metrics, Lisa can choose which properties of the query should be generalized or removed [20]. By removing in the query concepts related to date and event, the query becomes Sydney and Building, and Lisa finds 29 photos. Figure 4.3 shows the three selected photos in the initial query (left side), and the resulting view of the final query (right side). From there, she can further zoom-out, zoom-in to find photos of modern buildings, or shift to find buildings in different countries. *Interactive query relaxation* [136] is similar, except that only one facet is retained in the

$q = \text{Australie and not Portrait}$

$q = \text{Batiment and Sydney}$

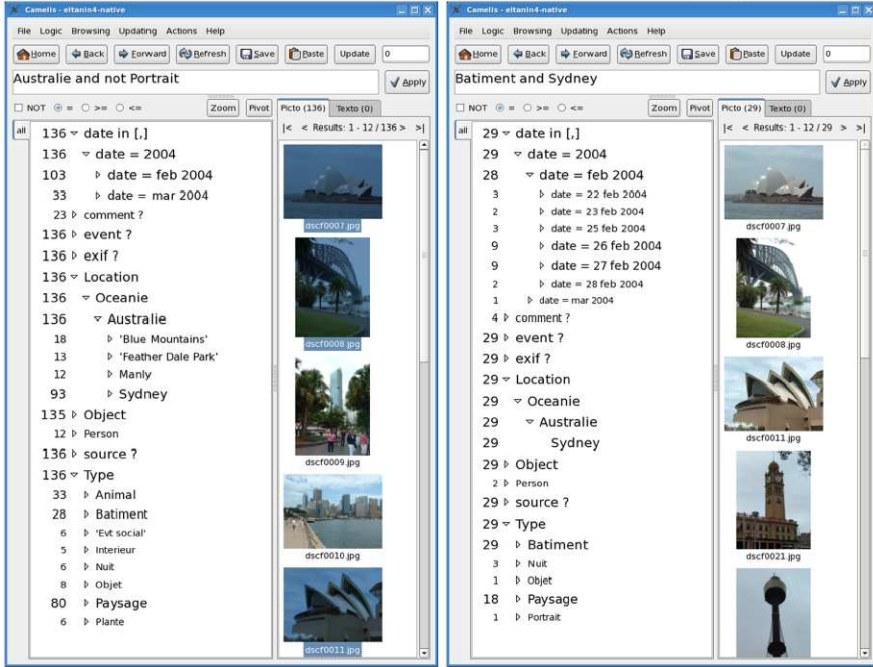


Fig. 4.3 A screenshot of CAMELIS before and after querying by examples

generalized query. For instance, starting with the same photos, Lisa could reach the query Sydney or the query Building, but not Sydney and Building.

A special case of querying by examples is when selecting only one photo. Then there is only one object in the extent, because there are enough properties to uniquely characterize each photo, and the query contains all the object properties, which are more easily read in the dynamic taxonomy. So this is an easy way to access the full description of any object.

4.4 Design Patterns

This section gives an overview of solutions for solving the issues and challenges in the user interface design of applications for faceted browsing and dynamic taxonomies, with a special focus on how to enable the previously introduced navigation modes. Where applicable, these are referred to the respective user interface design patterns from established pattern libraries.

Fig. 4.4 Mixing multi-select and single-select facets in the yelp (<http://yelp.com>) application

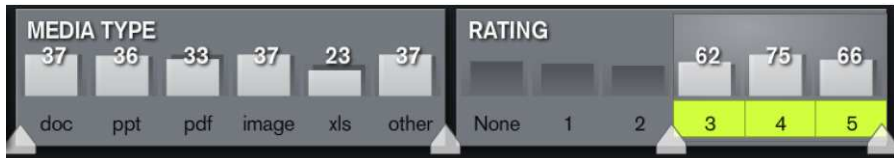


Fig. 4.5 The ContentLandscape application (see Sect. 4.5.3) combines bar chart representations with slider controls for range selection

4.4.1 Selection Management

Filter selection and de-selection is of central importance in faceted search. The basic navigation modes of *zoom-in* and *zoom-out* are present in all examined user interfaces.

If only one concept should be selectable at a time within a facet (thus avoiding the possible confusion if multiple values are to be connected by AND or OR), traditional single-select controls such as *radio buttons*, *dropdown list controls* or *simple links* (in web applications) are advisable. The standard multi-select elements, on the other hand, are *check boxes*.

Interfaces that allow only one concept selection per facet support *shift* navigation in the easiest manner, since only one click is necessary to replace a selection with another one from the same facet. If multiple selections are allowed per facet (*slice-and-dice* navigation mode), a distinction has to be made between *zooming-in*, adding the clicked value to the active filters, and *shift*, i.e. replacing the previously selected concepts from the same facet.

For instance, the yelp⁴ web application provides check buttons for multi-select facets and simple links for facets with exclusive selection (see Fig. 4.4). Alternatives for allowing both modes in a facet would be dedicated controls (e.g. a “jump to” button), or modifier keys (such as pressing “shift” while clicking).

For *range selection* navigation mode, *slider controls* can allow the specification of upper and lower bounds on the result set (see Fig. 4.5C).

Additional UI functionality, however, is usually accompanied by additional complexity and visual clutter. Intelligently limiting users’ options can help in allowing

⁴<http://yelp.com>.

Fig. 4.6 The Exhibit (<http://simile.mit.edu/exhibit>) user interface signalizes missing concept assignments in a facet



the user to focus on his core tasks without additional burden of rarely used functionality. For example, for a web shop application, it might be sufficient to split the “price facet” into 3–5 discrete regions from low- over mid-priced to expensive goods, instead of giving the more fine-grained option to filter from 37 to 82.

Either way, concept de-selection should be as easy as concept selection. Additionally, if breadcrumbs or a similar filter summary indicator are present, these should include the option to clear individual filters as well. Also, buttons for resetting single facets or all filter options can help to *zoom-out* quickly.

Pivoting is usually supported not directly in the facet panels, but from the detail views for single contents. First established in Web 2.0 applications [189], it has become a common practice that a metadata value clicked in the content presentation leads to a new view with the respective value as the only selected concept. The same holds for *Querying by examples*, as this action is intrinsically related to resource instances, and not to individual facet concepts. Consequently, querying by example is usually realized with context menus or buttons adjacent to the result list presentations or detail view.

If the data is only partially tagged, it is advisable to include a “no value assigned” concept, as for instance, demonstrated in the Exhibit prototype [145] (see Fig. 4.6).

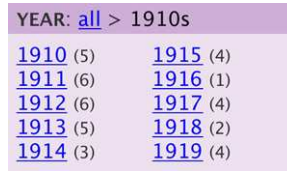
4.4.2 Revealing Hierarchy

For flat facets, i.e. not featuring a hierarchical relation between the concepts, simple list widgets are usually used. List sorting can either be alphabetical, or dynamically updated by the number of assigned items in the current result set. For navigating hierarchies, a number of different presentation and navigation options exist, which are discussed in the following.

4.4.2.1 Explorer Tree

The expandable explorer tree constitutes an established representation for hierarchical structures. This principle is, for example, used in the Camelis application (see Fig. 4.2C). Given the usually quite limited screen estate, however, the expanded lists often exceed the available facet widget space. This leads to the need for scrolling, which makes it difficult to orient in the hierarchical structure, especially if multiple levels are expanded.

Fig. 4.7 Zoom-and-replace and breadcrumbs in the Flamenco application



4.4.2.2 Zoom and Replace

The Flamenco application⁵ [327] zooms into selected values, replacing the facet widget content with the level below the selected concept. The path from the root of the hierarchy to the current level is accessible via breadcrumbs in the header of each facet widget (see Fig. 4.7). This pattern works only for single-select facets, and optimizes for item presentation on one level. Consequently, navigation across the tree is facilitated.

4.4.2.3 Collapsible Panels

The ContentLandscape application [279] features compact, hierarchical widgets based on the accordion pattern,⁶ where each hierarchy level is represented as an individual accordion level. On concept selection, the respective level is collapsed, and the subsequent level opened, to allow further drill-down in the hierarchy. Moreover, opening a level is possible by simply clicking the respective accordion header (see Fig. 4.8C).

4.4.2.4 Continuous Zooming

In the FacetZoom prototype⁷ [78], hierarchical facets are displayed as space-filling widgets which allow a fast traversal across all levels while simultaneously maintaining context (see Fig. 4.9). It supports both horizontal panning for exploring a whole hierarchy level, as well as *tap-and-center* navigation, allowing to dynamically zoom-in on tree nodes. For selected concepts, the child nodes are displayed on top of the widget. Navigating one level up the tree is supported by a bottom row presentation of the parent node.

⁵Online demos available at <http://flamenco.berkeley.edu/>.

⁶See e.g. <http://www.welie.com/patterns/showPattern.php?patternID=accordion>.

⁷Open source version available at <http://advancingusability.wordpress.com/2008/03/31/facetzoom-first-open-source-release/>.

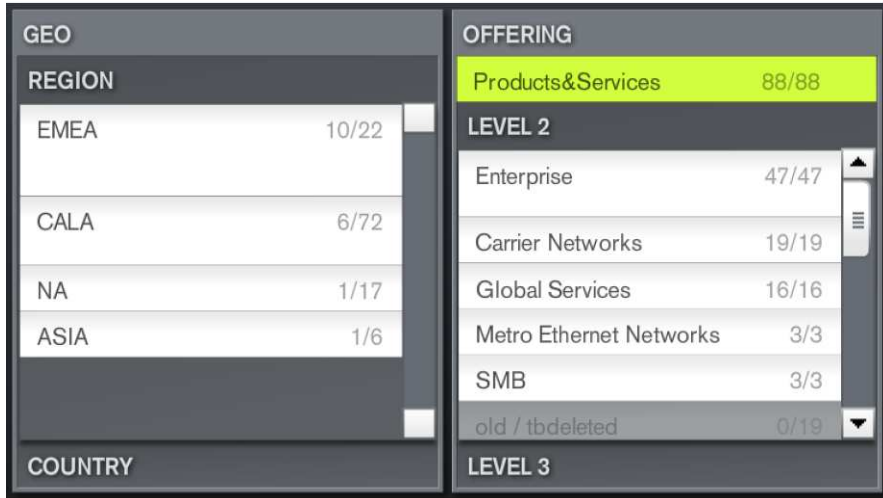


Fig. 4.8 The ContentLandscape application applies the collapsible panel pattern for zooming into concepts within a hierarchy

Fig. 4.9 The FacetZoom widget combines ideas from zoomable user interfaces (ZUIs) with faceted search



4.4.3 Facet Management

A variety of options to overcome the problem that, often, more facets are available than can be put on screen at the same time, are discussed in [131] and [133]. The options range from *collapsible facet widgets* (such as used by, for instance, Getty images’ faceted navigation interface⁸) over expandable filter areas (“More . . .” button) to dynamically selecting the shown facets based on the existing query (as demonstrated in the yelp application).

⁸<http://gettyimages.com>.

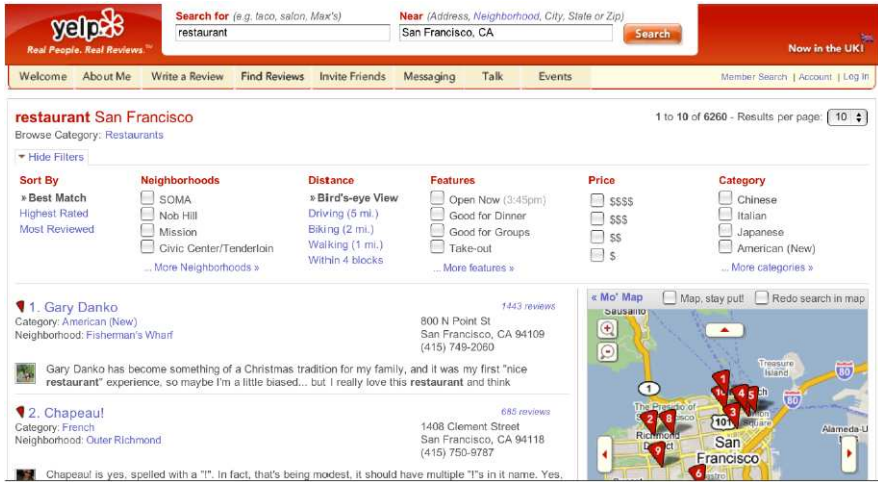


Fig. 4.10 The yelp application automatically selects the presented facets based on the search term

4.4.4 Keyword Search

As noted above, a free-form keyword field in order to search for arbitrary terms in addition to the defined classification scheme is a “key component to successful faceted search interfaces” [131]. This task is especially challenging, since a search field can either act as a filter on the resources, e.g. searching over titles and descriptions, or if it can also match classification terms. The following sections discuss further variations within these two options.

4.4.4.1 Keyword Search as Additional Resource Filter

When search engines are enhanced with faceted search, often, a keyword search is used to define the initial result set, which can be further refined by concept selections from facets. For instance, the yelp application (see Fig. 4.10) asks for a topic (e.g. “auto repair”) and a location (e.g. “San Francisco”) to be entered, before entering the faceted search mode. In this application, displayed facets and concepts are selected dynamically depending on the type of query, i.e. a search for “auto repair” will yield different filtering options than one for “Chinese restaurant”.

The Flamenco application [327] allows to choose between a full-text search over all results (overriding other filters) or within the current focus (see Fig. 4.11).

Obviously, when integrating keyword search with dynamic taxonomies, there might be zero hits, due to the unrestricted nature of the input. This violates the poka-yoke principle that we identified as one of the key features of applications in this domain. One solution could be to check for results after the user submits the keyword query, and leave the keyword filter in a “tentative” state if no results are

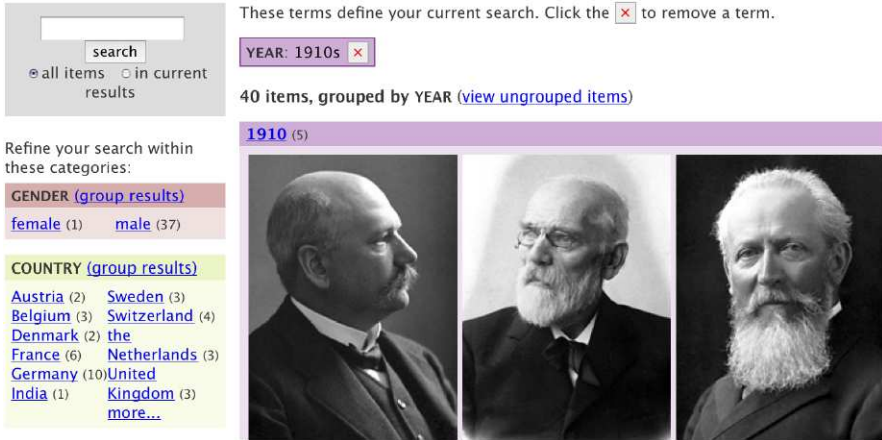


Fig. 4.11 The Flamenco application allows to choose between a full-text search over all results (overriding other filters) or within the current focus

found within the current focus. This would give the user the option to either zoom-out some other filters, or re-tract the query. In this case, it would be helpful to have a preview, of how many results could be achieved, if the respective concept would be removed from the query.

4.4.4.2 Keyword Search Within Facets

In order to avoid having to navigate large hierarchies, even though the target concept is already known by name, direct access to facet items can be achieved with a keyword search over the concept labels.

For instance, the /facet system [139], provides a keyword search box for each facet (see Fig. 4.12). This interface dynamically suggests matching concept labels after the user has typed a few characters; only keywords that produce actual results are suggested. This makes the interaction often faster than manually navigating the tree. The results are presented in a collapsible tree structure. Additionally, if the target concept is known, but it is unclear, in which facet it is located, a global search box executes the described operation over all search boxes in parallel.

A similar approach is described in [30], which is even extended to finding facets by label, and can thus be applied to very large and heterogeneous resource bases.

The ContentLandscape application [279] features a combo box component for quick access to concepts across all hierarchy levels (see Fig. 4.13C). It can be opened by clicking a search button in the facet box. In its initial state, the text field is empty, and a scrollable, alphabetical list presents all concept labels from that facet, regardless of depth level. When the user starts typing, this list is dynamically reduced to terms matching the query.

Fig. 4.12 The /facet system allows to quickly search within the concept labels of a facet



Fig. 4.13 Quick access to concepts with a combo box in the ContentLandscape application



4.4.5 Filter Summary and History Navigation

Breadcrumbs can be used to summarize the current selection status in one central place in the user interface. These usually record the sequence of selection actions across all facets [131]. Breadcrumb entries should be clickable, leading to a zoom-out action on the respective concept. The footnote web site combines breadcrumbs with the option to refine with an additional keyword search (see Fig. 4.14).

4.4.6 Animated Transitions

Animated transitions can facilitate awareness of transformations and responses in user interface design [135, 291]. Perception of change is especially important for facet browsing, as the sudden disappearance of list items after click can be a source for misconceptions and confusion. In fact, studies have shown that so-called *change blindness* [222] is a common psychological phenomenon: changing details of visual scenes are often remain unnoticed, if the two states are separated by a short flash, as it is common, for example, in web applications. The Elastic Lists facet browser [278] demonstrates how smooth transitions can help in understanding filtering processes.⁹

⁹Available at <http://moritz.stefaner.eu/projects/elastic-lists/>.

Fig. 4.14 The footnote web site combines a filter summary with the option to refine with an additional keyword search

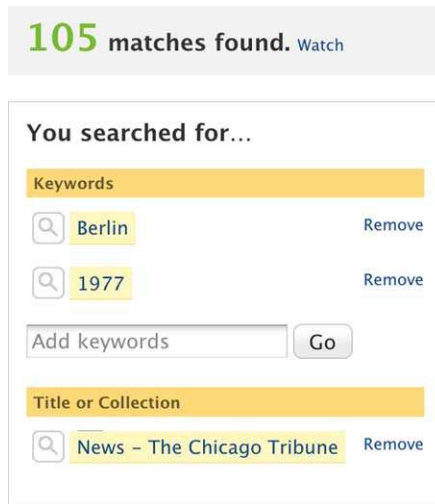


Fig. 4.15 The RAVE system visualizes metadata value proportions in horizontal bar charts

4.4.7 Visualizing Proportions

As stated above, one common and useful technique is to exclude concepts with zero occurrences from the presented filter options, in order to avoid selections with zero results. For exploratory tasks, it can be useful to additionally see *how many* items match each of the respective concepts in the given focus.

Besides support for orienteering (see Sect. 4.1), analyzing metadata distribution can constitute a valuable information source by itself, e.g. in order to understand what makes a data set special compared to the whole collection and to generate hypothesis about the underlying reasons.

The simplest option is to provide this information in brackets after the concept label (e.g. “Europe (5)”). While this presents an economic and easy solution, the user is left with the task of processing and understanding these numbers. Visualiza-



Fig. 4.16 Elastic lists indicate the number of matched resources in scaling list entry height. Additionally, unusually high proportions (compared to the global distribution) are indicated by brightness of the list entries

tion can help to make the relevant information available pre-attentively and in an intuitive manner.

For instance, the RAVE system [332] visualizes metadata proportions in horizontal bar charts, below the concept label (see Fig. 4.15). While the graphic layout could be improved in order to introduce less visual clutter, the prototype shows how additional information about local and global weights can be integrated without loss of screen estate.

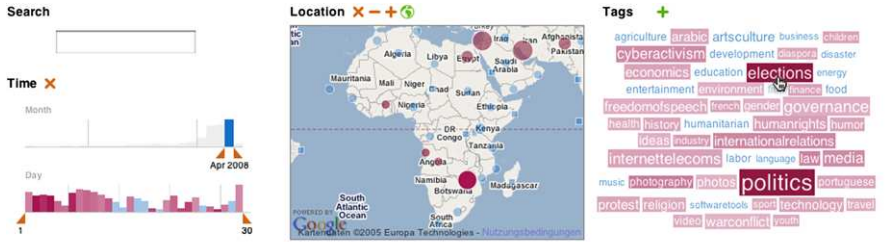
In the Elastic Lists prototype [278], the height of a list item indicates the relative proportion of items associated with the respective metadata value in the given context (see Fig. 4.16). Additionally, a brighter color indicates that the current weight is significantly higher when compared to the global distribution. List entries with a weight of zero (i.e. not occurring in the current context) are collapsed to a minimal visible height.

The Visgets system [91] extends this principle by featuring a whole number of visualizations, with a weighted, coordinated brushing scheme (see Fig. 4.17C). The visualization elements include bar charts with range sliders, a map, and a tag cloud.¹⁰ Visual representations for concepts and metadata values are scaled according to their global proportion. The coloring indicates, on the one hand, presence or absence of the respective value in the current result set. Additionally, on rollover on any concept or metadata value, more strongly associated items receive a higher opacity (*weighted brushing*).

4.5 Extensions and Related Approaches

Lately, the described principle of faceted search and dynamic taxonomies are being extended and translated to other types of search and browsing applications. This section provides some brief pointers to current research in this area, and introduces the novel principle of out-of-turn interaction.

¹⁰See <http://www.welie.com/patterns/showPattern.php?patternID=tag-cloud>.



Results 56 of 442



Fig. 4.17 Weighted, coordinated brushing in the Visgets system

4.5.1 FaThumb

FaThumb [156] enables faceted search on mobile devices (see Fig. 4.18C). The filter area is grouped in nine zones, corresponding to the nine digit keys on mobile phones. The middle zone serves as a spatial overview during navigation. The surrounding eight zones allow the user to select hierarchy branches and repeatedly zoom in on subtrees. The left short shortcut key adds the currently selected concept to the query, the right one allows to quickly jump back to the top.

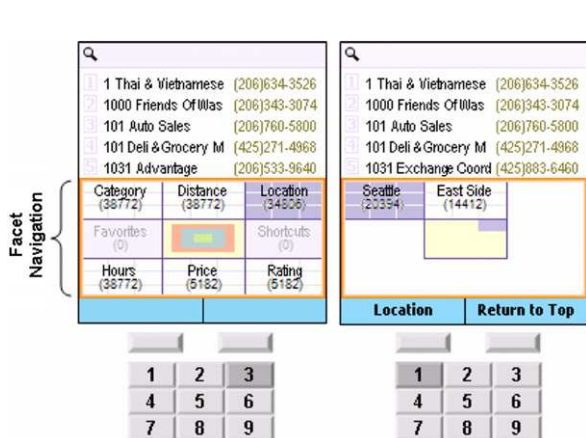


Fig. 4.18 Faceted search for small screens in the FaThumb prototype

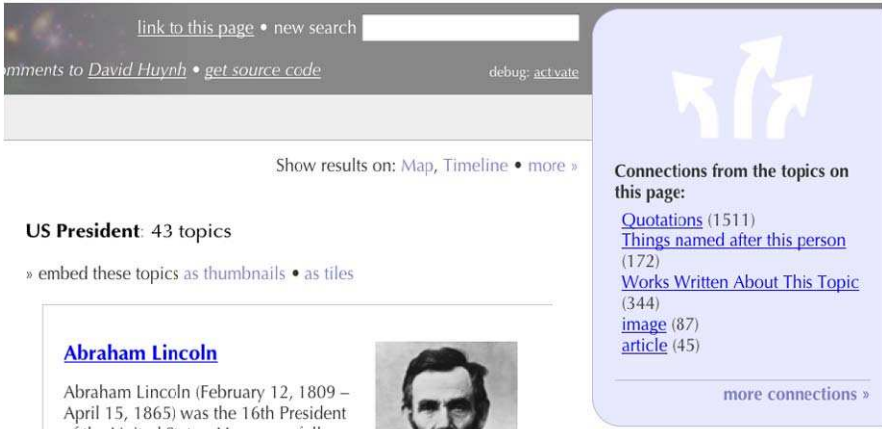


Fig. 4.19 The parallax application allows to jump to related sets of items from a faceted browsing situation

4.5.2 Browsing Related Entities

Usually, the type of resource entity to be browsed (e.g. book, car, web page ...) remains fixed in a faceted browsing application. In [159], a conceptual prototype of a browsing application named *Humboldt* is described, which allows to switch the type of displayed entities based on relations to the current result set. In principle, the application allows to treat any facet value space as a search result list, and arrange the interface accordingly. To cite an example given in [159], “a user who filters films on certain directors and then pivots¹¹ on actors will see all actors in the result list, who are related to any film in the previous result list”.

This principle has also been demonstrated in the parallax application based on the freebase¹² public database [144] (see Fig. 4.19). It allows, for example, to query the data set for architects, then filter down to all modern architects: a classical zoom-in. The novel principle, however, is that the user can explore related collections, like the buildings they designed, their birth places etc. in the same facet browsing space. The jump to these new results is offered in a “connections” box on the top right of the interface. History navigation for these “related set” browsing steps is provided by a breadcrumb control.

¹¹Note that the semantics of *pivoting* in this case differs from the definition introduced in Sect. 4.3.4.

¹²<http://freebase.com>.

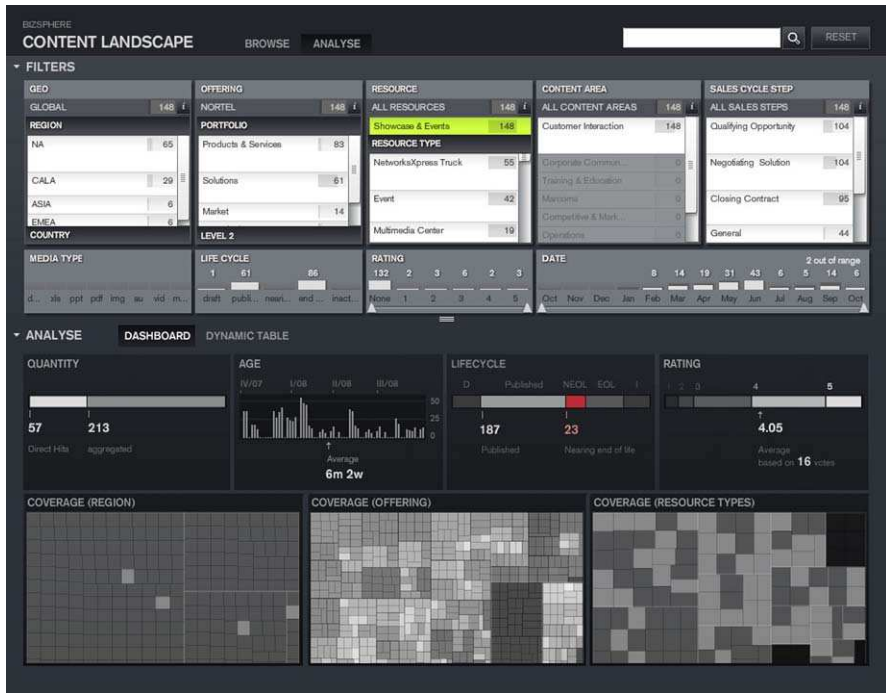


Fig. 4.20 The dashboard view of the ContentLandscape application

4.5.3 Resource Analytics

Understanding resource production, use and distribution across departments, regions, and product groups is one of the core challenges of knowledge management in the enterprise [230]. “What are the most downloaded contents?”, “do the presentation materials for a given product cover all important sales regions?”, “what parts of my resource collection are growing? and which are declining?” are typical questions in this area.

The ContentLandscape application¹³ [279] is part of the BizSphere¹⁴ application suite and uses faceted browsing and search in order to facilitate the understanding of resource distributions. In addition to traditional result set views, a dashboard view presents statistical measures for the resource set in the current selection (see Fig. 4.20C). It features visualizations of trend measures such as the quarter to quarter growth, a detailed age histogram, and the rating distribution. Moreover, the coverage of the selected resource set with respect to the three main taxonomies ‘region’, ‘offering’ and ‘resource type’ is presented in squarified treemaps [52, 270]. At first

¹³<http://moritz.stefaner.eu/projects/content-landscape>.

¹⁴<http://bizsphere.com>.

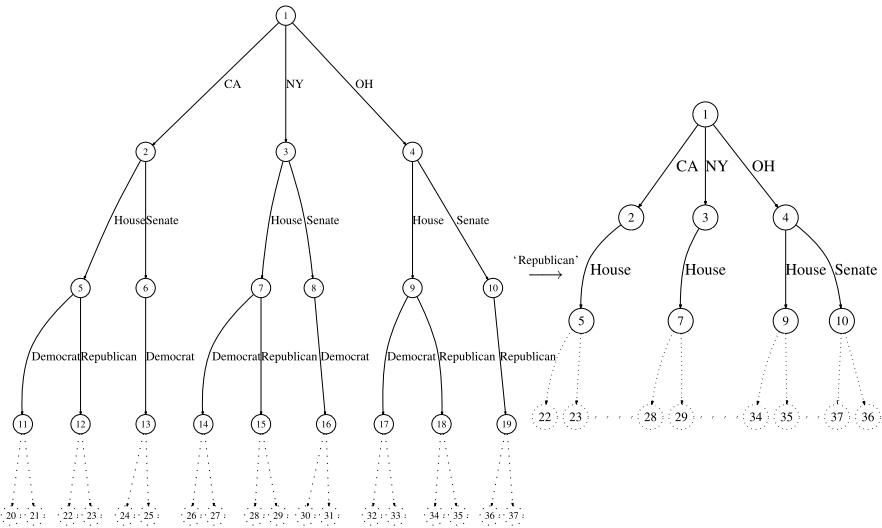


Fig. 4.21 Illustration of the pruning conducted as a result of out-of-turn interaction. (left) Sample hierarchy, simplified for purposes of presentation, with characteristics similar to those in PVS. (right) Site schema resulting from supplying ‘Republican’ out-of-turn

glance, this visual display allows to see, for instance, if all product groups are represented by resources in the current selection, and in which specificity. This statistical analysis can be further decomposed into in several matrix views. Inspired by OLAP approaches [70], these allow the user to split the result set according to up to three dimensions, and compare the statistical measures for the resulting sub-collections in parallel.

4.5.4 Out-of-turn Interaction

Out-of-turn interaction [195] is a technique for navigating hierarchical websites which augments traditional browsing by empowering the user to supply a hyperlink label which is presented beyond the current webpage (hence out-of-turn) to initiate a search over the site’s hierarchical schema. When the system receives an out-of-turn input, it removes all paths through the site which do not contain a hyperlink labeled with the input and removes the hyperlink labeled with the input from the remaining paths. Figure 4.21 illustrates how a sample hierarchy with a structure similar to that of PVS would be pruned based on supplying ‘Republican’ out-of-turn. Notice that all paths leading to the webpages of Democratic politicians (nodes 20, 21, 24, 25, 26, 27, 30, 31, 32, and 33) have been removed. In addition, the hyperlinks labeled ‘Republican’ in the remaining paths (those leading to nodes 22, 23, 28, 29, 34, 35, 36, and 37) have been removed.

Figure 4.22 illustrates an out-of-turn interaction through a browser toolbar we call *Extempore* (as it permits the user to supply terms *extemporaneously*). Here the



Fig. 4.22 Illustration of an out-of-turn interaction with PVS through the *Extempore* toolbar

user supplies ‘Republican’ out-of-turn (see Fig. 4.22, top). This causes some of the hyperlinks presented on the root page (e.g., Hawaii), those which do not lead to the webpages of Republican congresspeople, to be pruned out (see Fig. 4.22, bottom). When used in conjunction with traditional browsing, the unsolicited reporting [19] involved in supplying an out-of-turn input supports a simple form of mixed-initiative interaction [211] and can be viewed as an approach to integrating querying and browsing in information hierarchies [45].

In sites where each level of the hierarchy corresponds to a facet of information assessment, such as PVS, out-of-turn interaction permits the user to explore the facets in any order without the designer enumerating all possible paths of navigation. In hierarchies where each level does not correspond to a facet, such as Yahoo! and the Open Directory Project (ODP) at dmoz.org, out-of-turn interaction behaves more as a pruning operator and reveals to the user the portions of the taxonomy pertaining to their query. For example, ODP contains 16 top-level categories and a user starting from the homepage would be hard-pressed to know that only four (Home, Shopping,

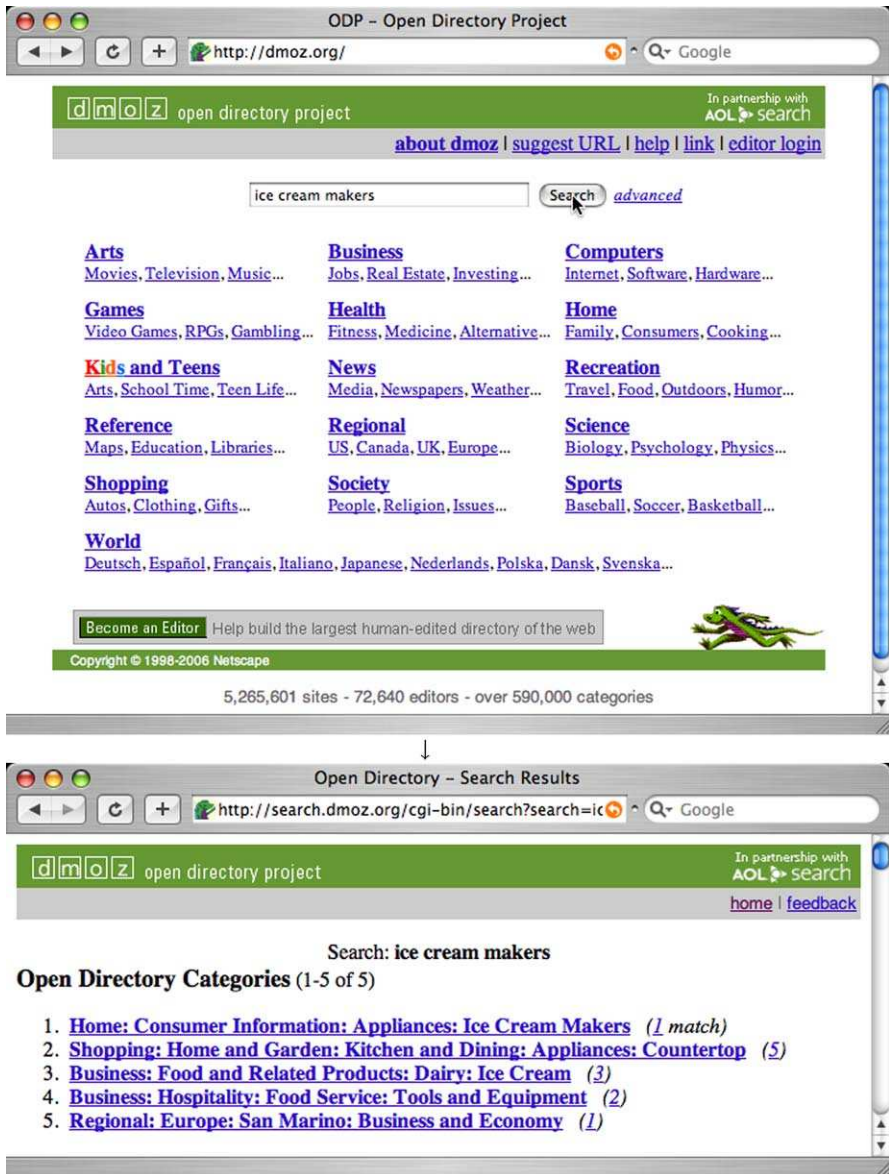


Fig. 4.23 (top) A query for ‘ice cream makers’ in the Open Directory Project and (bottom) its result as a flat list

Business, and Regional) contain links to information about ‘ice cream makers’. An out-of-turn interaction reveals these categories. In fact, the search feature provided in ODP is similar to out-of-turn interaction with the exception that ODP flattens

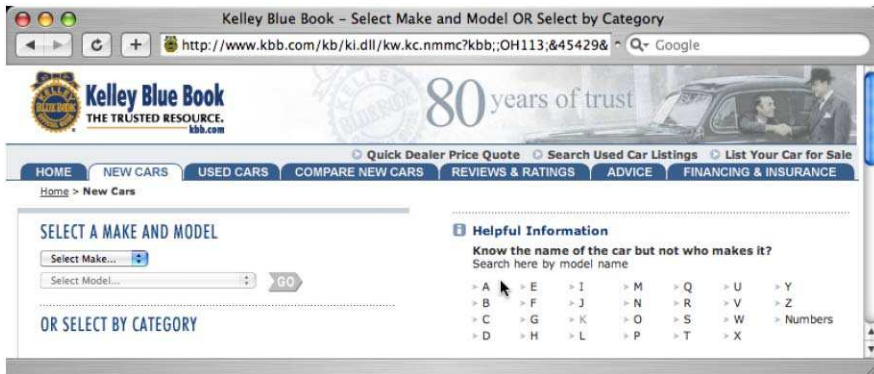


Fig. 4.24 Facility for automobile-make lookup by model in the online Kelley Blue Book

the hierarchical structure in response to a query (see Fig. 4.23) whereas out-of-turn interaction preserves the hierarchical nature in order to retain context.

Notice that with the interpretation of out-of-turn interaction presented here, an unexpanded query will yield the same result as its expanded version and therefore query expansion here is simply a feedback mechanism to expose dependencies, unlike its use in traditional IR.

There are other means of exposing dependencies underlying information hierarchies during information-seeking. For example, the Kelley Blue Book (KBB) online at kbb.com provides a facility for automobile-make lookup by model (see under heading titled ‘Helpful Information’ in Fig. 4.24) since FDs of the form ‘*model* → *make*’ are implicit in the domain of automobiles. When browsing new cars in KBB, users are first asked to make a selection for automobile make (see under heading titled ‘SELECT A MAKE AND MODEL’ in Fig. 4.24). The lookup facility allows the user to search for the make of an automobile based on the model so that they can proceed with the information-gathering dialog on the left side of the window in Fig. 4.24. A more sophisticated example of support for dependency exploration is *Sony’s Advisor* facility available through sonystyle.com when browsing products such as digital cameras and camcorders.

4.6 Personalizing Faceted Search

4.6.1 Introduction

One of the primary ways users manipulate a faceted search interface is by refining their current query by clicking a facet-value pair from a list of possible system suggestions. How effective users are at finding their documents of interest is related to the quality of the query refinements suggestions. Traditionally, ad hoc approaches have been used to determine which values for a facet should be presented to the

user during interaction. One common approach is to simply display all available values for a facet. While this may be effective when the number of available values is small, this approach may overwhelm users when the number too large [271]. Another approach is to display only the first few alphabetically ordered values [134]. While this approach avoids overwhelming the user with many values, it arbitrarily biases the interface towards values earlier in the alphabet. A better method is to display the most frequent values for a facet. However this method is not user centric since the most frequent values are endemic to the corpus instead of the users.

This chapter focuses on a user centric approach to determine which values are most useful to users: personalizing faceted search. Personalization allows the system to present the facet–value pairs that can help the user quickly find the document(s) that the current user is most interested. In order to determine which facet–values are most useful to a particular user, we analyze the distribution of values in corpus, and user’s feedback on documents while using the system. With this knowledge, we can tailor the faceted search interfaces to individual users.

4.6.2 Related Work: Personalized Search and Filtering

The idea of personalizing search is not new in the information retrieval community [16, 41, 53, 74, 108, 140, 153, 268, 285, 289, 320]. Dumais, Cutrell, Sarin and Horvitz automatically generate queries based on keywords within an email being read or composed by a user [94]. To improve retrieval results, Bharat treats the previous information requests from the user as the context of the current query [41], whereas Shen, Tan and Zhai use the preceding queries and clicked-document summaries as the context of the current query [268]. On the other hand, researchers have developed personal information integration environments that store a particular individual’s heterogeneous information and the context of the information, providing content and context-based retrieval [14, 89, 117]. Rui et al. [170] explored biasing cosine similarity methods based on user feedback in order to retrieve more documents that were similar to a user’s interests. Abrol et al. personalized semi-structured search interfaces by using a user’s transactional feedback from his/her queries [12]. Shen et al. used implicit user feedback, such as query refinement and click logs, to customize a KL-divergence model for document retrieval [268]. Personal WebWatcher passively observed a user’s browsing behavior in order to highlight links that matched the inferred task [192].

On the other hand, personalization is a heavily studied problem in the information filtering research community and the research can be traced back to 1970s. For example, *content-based adaptive filtering* studies the scenario in which a recommendation system monitors a document stream and pushes documents that match a user profile to the corresponding user. The user may read the delivered documents and provide explicit relevance feedback, which the filtering system then uses to update the user’s profile using relevance feedback retrieval models or machine

learning algorithms (e.g. Boolean models, vector space models, traditional probabilistic models [229], inference networks [55], language models [75], Support Vector Machines, K -nearest neighbors clustering, neural networks, logistic regression, or Winnow [167, 324]). *Collaborative filtering* goes beyond merely using document content to recommend items to a user by leveraging information from other users with similar tastes and preferences. Memory-based heuristics and model-based approaches have been used [15, 36, 49, 85, 138, 141, 154, 160].

4.6.3 Personalization Based on Collaborative Filtering

Faceted search interfaces share three characteristics. The interfaces present a number of facets along with a selection of their associated values, any previous search results, and the current query.¹⁵ By choosing from the suggested values of these facets, a user can interactively refine the query. The interface also provides a mechanism to remove previously chosen facets, thus widening the current search space.

In personalized faceted search, the key problem is to rank facet–value pairs according to how helpful they are for a particular user to use for query refinement. Complicating this task is the fact that rarely, if ever, does the system have access to user ratings of individual facet–value pairs. Instead, most of the existing faceted search systems, such as Amazon.com and Netflix.com, are designed so that users rate each individual document. This design trade-off is reasonable since the relevance/rating of a facet–value pair is not a well defined problem and thus hard for the user to provide. Besides, a user usually has seen a overwhelming number of individual facet–value pairs, and users rarely experience any particular facet–value pair in isolation. By rating the whole document, a user express a preference over many facet–value pairs simultaneously, especially for facet–value pairs that they may only be tangentially aware of.

Personalized faceted search is a comparatively new field that has not been well studied. Fortunately, we can develop personalized faceted search interface based on the earlier work in personalized search. For example, we can first use traditional collaborative filtering techniques to predict a user’s ratings on unseen documents, and then propagate information from ratings on whole documents to individual facet–value pairs.

The basic assumption of collaborative filtering is that users that have similar preferences on some documents may also have similar preferences on other documents. Therefore the algorithm provides recommendations for a user based on the opinions of other like-minded users.

In collaborative filtering, users ratings over items are represented as a matrix A , where $A_{u,i}$ is user u ’s rating on item i . Many collaborative filtering techniques have

¹⁵For an interactive faceted search interface, the current query is the facet–value pairs the user has selected so far.

been proposed to predict the missing cells in the matrix [15, 36, 49, 85, 138, 141, 154, 160].

In this chapter, we first introduce two basic collaborative filtering techniques: K nearest neighbor and singular value decomposition. We choose the two techniques because they are commonly used, very different from each other and are complementary to each other. Variations of these two techniques have been successfully used in the Netflix movie recommendation competition [36]. Then we discuss how to go from document ratings to facet–value pair ranking.

4.6.4 K -Nearest Neighbors Based on Item–Item Similarity or User–User Similarity

There are two very commonly used collaborative filtering approaches: the first one compares each user to the other users, while the second one compares the items to each other. These are called *user–user similarity* and *item–item similarity* respectively. We describe Konstan et al.’s Pearson’s correlation based user–user algorithm [160] below.

For each user, we calculate the average rating assigned by that user \bar{A}_u to all rated items. Each unknown rating is then estimated as the user’s average rating, perturbed by sum of the difference between every other user’s assigned rating and his/her average rating, weighted by the correlation among the commonly rated items of the current user to every other user.

Formally, this is stated:

$$A_{u,i} = \bar{A}_u + \sum_{v=1}^k \frac{w_{u,v}(A_{v,i} - \bar{A}_v)}{|w_{u,v}|} \quad (4.1)$$

where k is the number of users that have at least one rated item in common with user u , and $w_{u,v}$ is the Pearson’s correlation between user u ’s ratings and user v ’s ratings. Recall that Pearson’s correlation is:

$$\begin{aligned} w_{u,v} &= \frac{\sum_{j=1}^m ((A_{v,j} - \bar{A}_v)(A_{u,j} - \bar{A}_u))}{\sigma_v \sigma_u} \\ &= \frac{\sum_{j=1}^m ((A_{v,j} - \bar{A}_v)(A_{u,j} - \bar{A}_u))}{\sqrt{(\sum_{j=1}^m (A_{v,j} - \bar{A}_v)^2)(\sum_{j=1}^m (A_{u,j} - \bar{A}_u)^2)}} \end{aligned} \quad (4.2)$$

where σ_u and σ_v are the standard deviations in user u ’s and user v ’s ratings, and m is the number items that user u and user v have both rated.

Herlocker et al. [138] examined using other similarity methods such as Spearman’s correlation, information entropy, mean-squared difference, and found they performed similar to Pearson’s correlation.

In its simplest form, the item-item algorithm is similar to the user-user algorithm, only with the rows and columns exchanged. Item-item similarity can be extended to take into account the content of the items being rated. For example, Sawar et al. [263] estimated ratings by summing the ratings of the other rated items, weighted by the cosine similarity of the rated and unrated plain text documents.

4.6.5 Singular Value Decomposition

One problem with the collaborative filtering techniques discussed in Sect. 4.6.4 is that the user–item matrix can become very sparse as the number of items and users increase. This sparseness can sometimes lead to poor predictions. By creating a low dimensional approximation of the rating matrix, it is possible to improve the accuracy of the predicted ratings. One such technique that has been used successfully in the personalization and collaborative filtering domains is *singular value decomposition* (SVD) [36, 113, 209, 262, 264].

SVD, also known as *latent semantic indexing* (LSI) in the information retrieval community [83], works by combining rows and columns that are found to be strong correspondence. These correspondences are called *latent factors*.

Applying SVD to collaborative filtering task, we factor the $m \times n$ user–item matrix A into three smaller matrices U , Σ , and V . U is $m \times h$, Σ is an $h \times k$ diagonal matrix, and V is $n \times h$ matrix.

$$A \simeq U \Sigma V^T \quad (4.3)$$

The values along the main diagonal of Σ are the biggest h *singular values* of A in decreasing order. Each row of U and V contain orthogonal *singular vectors*. The vectors in U are known as the *left singular vectors*, while the vectors in V are the *right singular vectors*. Since we want a low dimensional approximation of the rating matrix, only the first $h < \text{rank}(A)$ singular values and singular vectors are used.

Typically, A is approximated as the product of two matrices:

$$A \approx (U(\sqrt{\Sigma})^T)((\sqrt{\Sigma})^T V) \quad (4.4)$$

We can view the first matrix as the hidden representation of the users, and the second matrix as the hidden representation of the items. With this approximation, the predicted rating for a user u on item i can be calculated as $A_{u,i} = \bar{A}_u + (U(\sqrt{\Sigma})^T)_u^T ((\sqrt{\Sigma})^T V)_i$.

4.6.5.1 Recommending Facet–Value Pairs

There are many methods to propagate information from ratings on whole documents to individual facet–value pairs. One method is to assign facet–value pair x_i a score

based on the expected rating a user u gives to a document d containing that facet–value pair:

$$\begin{aligned}
 f(u, x_i) &= E[R_u(d)|x_i \in d] \\
 &= \sum_{d \in D} R_u(d) P(x_i|d) \\
 &\approx \frac{\sum_{d \in D} R_u(d) I_{x_i \in d}}{\sum_{d \in D} I_{x_i \in d}} \tag{4.5}
 \end{aligned}$$

where D is the set of documents selected by the current query, $R_u(d)$ is the rating of user u on document d , and I is the indicator function.

One simple way to suggest facet–value pairs for query refinement is presenting the top scoring facet–value pairs that are contained in the documents selected by the current query. While this is an attractive option, this approach can suffer when the top scoring values are redundant, where an extreme case is that if they co-occur in the same documents. One elegant solution proposed by Chen and Karger [63] is to condition each suggestion on the assumption that none of the previous suggestions are relevant to the current query. For our purposes, this means that the $(k + 1)$ th suggestion is the top scoring value that is not contained in a document that contains any of the previous k suggested values.

To determine the order that the facets are presented to each user, a simple approach is using the average score of the suggested values for each facet for the null query and then fix order of the facets throughout the lifetime of the user interaction session(s).

4.6.6 Personalization Using Content Based Filtering

In this section we provide a complementary approach to personalization based on the content of the documents. Motivated by relevance feedback retrieval models and content based adaptive filtering techniques, we focus on two statistical models: a model of the documents being searched, and a model of a user.

4.6.6.1 Document Model

While every faceted document has a set of facets associated with it, the number of values that each facet has in a particular document can vary a lot. We model this by expressing the number of values each facet has in a random document as a draw from multivariate Normal distribution:

$$\langle n_1, \dots, n_K \rangle \sim MVN(\vec{\mu}, \Sigma) \tag{4.6}$$

where n_k contains the number of values for facet k in the document, $\vec{\mu}$ is a K -dimensional vector containing the mean number values for each facet, and Σ is the corresponding $K \times K$ covariance matrix.

Each facet in a document has a certain semantic meaning that dictates the type and thus the probability distribution of the values that can be associated with it. Five common types of facets are: *nominal*, *ordinal*, *interval*, *ratio*, and *free-text*. Nominal facets take discrete and orderless values. The values to this type of facet can be modeled as draws from a multivariate Bernoulli distribution. Ordinal facets also take discrete values, but there is an implicit ordering to these values. An example of this would be field that identified the sensitivity of a document as being for “full release”, “limited release”, or “secret”. Interval facets can take any value on a defined range, as long as the range excludes an explicit zero point. Ratio typed facets on the other hand can take zero as a value. Values for ordinal, interval, and ratio facets can be modeled as draws from a normal distribution. Free-text facets allow arbitrary text to be associated with documents. Traditional statistical information retrieval techniques represent each word in unstructured text as a draw from a multinomial distribution.

After identifying the type and the proper probability distribution over the values associated with facet, the probability of a document existing is simply the product of the probability of each desired value occurring for the appropriate facet.

4.6.6.2 User Model

Instead of estimating a user rating for a document, in this approach we estimate the probability of a document being relevant to a particular user. Similarly, instead of calculating a suitability score for a facet–value pair to a user, we estimate the probability of a particular facet–value pair appearing in a document relevant to a user. For simplicity and without loss of generality, we assume that a document is either *relevant* or *nonrelevant* to a user. From this we can estimate the probability that any document will be relevant to a particular user, and the probability that a particular facet–value pair x_k will be contained in a relevant or a nonrelevant document. This tuple is the user relevance model and is represented as:

$$\theta_u = \{P(\text{rel} | u), P(x_k | \text{rel}, u)P(x_k | \text{non}, u)\} \quad (4.7)$$

where $k = 1, \dots, K$.

These individual probabilities can be estimated from training data. For example, assume for a particular user u , there exists a set of training data $D_u = \langle D_{u,\text{rel}}, D_{u,\text{non}} \rangle$, where $D_{u,\text{rel}}$ is the set of documents marked by u as being relevant, and $D_{u,\text{non}}$ are the set of documents marked as nonrelevant. If the facet type is free text, the maximum likelihood estimation of θ_u is:

$$P(\text{rel} | u) = \frac{|D_{u,\text{rel}}|}{|D_u|} \quad (4.8)$$

Table 4.1 Facet types and distributions

Facet type	Values	Example	Distribution	Prior
Nominal	Unique tokens	Director	Multivariate Bernoulli	Multivariate Gamma
Ordinal	Repeatable ordered tokens	Critic’s rating (e.g. A, B, C, ...)	Normal	Normal
Interval	Repeatable numbers excluding zero	Year of release	Normal	Normal
Ratio	Repeatable numbers including zero	Running time	Normal	Normal
Free-text	Repeatable tokens	Synopsis	Multinomial	Dirichlet

$$P(x_k | rel, u) = \frac{1}{|D_{u,rel}|} \sum_{d \in D_{u,rel}} I_{x_k \in d} \quad (4.9)$$

$$P(x_k | non, u) = \frac{1}{|D_{u,non}|} \sum_{d \in D_{u,non}} I_{x_k \in d} \quad (4.10)$$

This setup is very similar to the commonly used relevance language model in information retrieval [331]. We leave it as an exercise for the readers to derive the estimation for other facet types.

As stated in Sect. 4.6.1, it may take a while before enough user specific feedback information is gathered from a particular user, thus user could suffer from the so called “cold start” problem. To handle this problem and get a level of acceptable performance from the very beginning, a hierarchical Bayesian model is used and found success in user modeling experiments [328, 334, 336]. In this model, each individual user model is considered as a draw from a prior distribution that is common to all users. By using common prior, gaps in a particular user’s model can be filled in by using information from the community of users. To applying the hierarchical Bayesian modeling approach for personalized faceted search, each distribution for each facet needs a separate prior that is estimated from the training data of all users in the system. Table 4.1 suggests different priors for different facet types.

Based on document models and user models described above, one can generate faceted search interface in various ways. For example, we can rank the facet–value pair based on (4.9).

4.6.7 An Ontological Approach

An alternative approach to personalize faceted search is using ontology created manually or automatically. Tvarožek and Bieliková [290] use the distance between values in a hierarchical ontology to measure similarity, and thus relevance to users.

In cases where an explicit ontology does not exist, one can be automatically constructed [42, 282, 288, 323].

The technique works as follows: Let $L_u(x)$ be the relevance of a facet x to user $u \in \mathcal{U}$ as computed by the ontological similarity of the facet and the user model [22]. In this case, the ontological similarity is the reciprocal of the maximum number of links needed to traverse from each value being compared, to a value common to both.

Each user model is then compared to the other users in the system, in order to calculate the *cross relevance* of a facet x to a user. This value, $C_u(x)$, is the average of the relevance of the facet to each user, weighted by the similarity of the each user v to the current user u .

$$C_u(x) = \frac{\sum_{v \in \mathcal{U}} \text{similarity}(u, v) L_v(x)}{|\mathcal{U}|} \quad (4.11)$$

The *global relevance* of a facet, $G(x)$, is calculated as the average relevance of the facet to every user. The *static relevance* of facet to a user is a weighted combination of the cross relevance and the global relevance.

In order to make recommendation to a user regarding a specific query, the temporary in-session relevance of each facet x to the current user u is introduced. This value, $T_u(x)$, is simply the fraction of query refinements (i.e. clicks) that utilize a facet x of all refinements in the current search session. This number is combined with the static relevance of the facet to determine the current *dynamic relevance* of a facet to the user. Values from facets with the highest dynamic relevance are then suggested to the user for query refinement. Evaluated with the Factic system, the ontology based approach reduce the number actions required for users to find their documents of interest when compared to an un-personalized baseline [290].

4.6.8 Evaluation Regime

Considering various personalized faceted search techniques, which one works better on a particular task? To compare personalization methods, an evaluation metric is needed. Traditionally user studies have been used to determine satisfaction with different user interfaces. While undeniably useful, user studies have some drawbacks. First, they are expensive to hold. A number of users must be gathered and then tested on the proposed system. This takes a nontrivial amount of time for even user studies with a moderate number of subjects. User studies also problematic when being used to evaluate personalized systems, as the test subjects may not interact with the system long enough for a sufficient user profile to be learned. This can lead to inconclusive, or possibly even contradictory results.

Koren et al. [162] proposed a complementary inexpensive evaluation metric based on calculating the expected utility to a user of a faceted search interface, through the use of simulated user interactions. This method allows designers to quickly compare various algorithms and determine which algorithms are the most

promising. By using this method, or a similar one, designers can conduct fewer user studies by only submitting the top performing algorithms for an in depth user study. A similar approach has seen success when evaluating spoken dialog systems [118, 166].

The evaluation system works as follows: Assume that the goal of the search interface is to enable users to find their documents of interest with the least amount of effort. In order to measure this effort, the actions that a user can perform when interacting with the system are identified and the system is rewarded or penalized depending on what action is performed. A series of user interactions are simulated using a combination of real-user feedback and heuristics. The interface is then scored according to the expected total reward for an interaction session.

Given S user interaction sessions, the empirical utility of the interface can be estimated easily:

$$U = \sum_s^S \sum_t^{T_s} R(q_{s,t+1}, a_{s,t}, q_{s,t}) \quad (4.12)$$

where $R(q_{s,t+1}, a_{s,t}, q_{s,t})$ is the reward the system receives if the user s takes an action $a_{s,t}$ at time t , which changes the query from $q_{s,t}$ to $q_{s,t+1}$.

Let us now define the reward function. As stated earlier, we assume that the goal of the interface is to allow users with the least amount of effort on their part. In order to measure this effort, the designer must identify which actions a user can perform at each step of the interaction. Koren et al. identified eight actions common to many personalized faceted search interfaces, along with the rewards the system receives for each. These actions are listed in Table 4.2.

Instead of employing real users through a user study, actions are simulated based on certain assumptions about how real users interact with faceted search systems. Without loss of generality, it is assumed that each simulated user is searching for exactly one *target document* and that the simulated user can recognize the document and the facet–value pairs that are indicative of that document. At each step of the search session, the simulated user scans the top ranked documents that match current query looking for the target document. If it is found, then it is selected and the

Table 4.2 User actions and rewards

Action	Reward
Select facet–value pair	negative
De-select user selected facet–value pair	zero
De-select system selected facet–value pair	negative
View more facet–value pairs	negative
Mark document as relevant	positive
Mark document as non-relevant	negative
View more documents	negative
End session	zero

session ends. If the target document is not found, the simulated user removes any facet–value pairs that are contained in the current query that do not match the target document. Once the query is cleaned of incorrect terms, the simulated user scans the list of presented facet–value pairs. A pair is selected by some method for inclusion in the query from the set of pairs that match the target document. If none of the suggested facet–value pairs match the target document, then a facet is chosen at random and all of its values are examined until a matching facet–value pair is found for inclusion. If no matching value can be found for any facet, then the simulated user scans through the complete list of returned documents until the target document is found.

When there are multiple matching facet–value pairs, deciding which one to include in the query can greatly impact how much additional searching is required to find the target document. Koren et al. suggested four possible selection methods. *Stochastic users* simply select one of the matching facet–value pairs randomly from a uniform distribution. *First-match users* scan the list from top to bottom, and select the first matching facet–value pair found. This heuristic is modeled after how users select matching documents from a ranked list of results. *Myopic users* select the matching facet–value pair that is contained in the least number of documents. With this method assumes that users are trying to reduce the search space as quickly as possible. *Optimal users* that perform actions that directly optimize the utility of interface were also identified, but not examined in detail due to the complexity in searching for the optimal policy for the user to execute.

Although the simulated users differ from real users, the evaluation methodology does provide insight into understanding how various faceted interface design algorithms perform [162]. This evaluation method is neither better or worse than user studies. Instead, the approach serves to complement user studies by being cheap, repeatable, and controllable.

4.6.9 Conclusions

This section presented the problem of determining which facet–value pairs the system interface should provide to a user for query refinement. In particular, we focus on personalized faceted search techniques that try to find facet–value pairs most useful to individual users. We introduced three major approaches, collaborative filtering based faceted search personalization, content based personalization, and ontology based personalization. We present a utility based evaluation framework for various faceted search interfaces, and the general idea is that the best interface should minimize the number/cost of interactions needed to find a document of interest. This general evaluation framework can applied to all kinds of facets, including facets whose values are organized as taxonomies.

Chapter 5

Extensions to the Model

Giovanni Maria Sacco and Sébastien Ferré

“The important thing is not to stop questioning.”

Albert Einstein, 1879–1955

This chapter describes the principal extensions to the dynamic taxonomy model, which include:

- data-mining functionalities in dynamic taxonomy exploration, which support both a sophisticated visual environment for data-mining applications, and wisdom-seeking exploration tasks;
- the extension to non-atomic, structured objects (such as video material) for which access at different levels of granularity must be supported;
- the virtualization of parts of the taxonomy, which are dynamically computed rather than explicitly defined. These extensions (virtual concepts, logics, and web ontologies) increasingly extend the base model from the expressive point of view, by dealing with an infinite number of concepts (such as those which represent numeric or date values), to concepts that are logical formulas, and to taxonomy shapes and extensions defined through logical inference;
- fuzzy dynamic taxonomies, in which the membership of object o to concept C is represented by a probability. Fuzzy dynamic taxonomies allow to model several practical situations (e.g., the frequency of a specific symptom in a pathology for medical diagnosis applications) in a more precise way, while retaining the ease of use and the normal operations supported by DTs.

In Sect. 5.7, we describe a number of useful techniques which use some of the features of DTs in a non-obvious way. These include:

- using predefined foci in order to personalize or control access;
- embedding SAES facets in dynamic taxonomies. We consider tag clouds, geographic maps and object clusters;
- accounting for popularity, recommendations, and authoritativeness by specific facets or taxonomic ordering;
- using a DT as a thesaurus to expand the IR index for a textual object in order to improve IR recall.

5.1 Data Mining

In this section we present an extended model [232, 252] that combines dynamic taxonomies and association rules in order to support interactive and guided data mining capabilities, with traditional measures such as support and confidence [18] and measures of statistical significance of associations. This extended model can be used for user-centered data mining in alternative to traditional approaches, but, most importantly, it also increases the user understanding of inner laws of the information base in normal applications, and can be especially important in wisdom-seeking tasks.

5.1.1 Association Rule Mining

Association rules can be formally defined as follows [18]:

Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of m distinct literals, called items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. We say that a transaction T contains X , a set of some items in I , if $X \subseteq T$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$.

In an association rule $X \Rightarrow Y$, X is called the *antecedent*, and Y is called the *consequent* of the rule.

The traditional (and minimal) measures for association rules are *support* and *confidence* [18]:

- *Support*($X \Rightarrow Y$) measures the “coverage” of the association rule, i.e., the probability $P(XY)$ that the rule holds.
- *Confidence*($X \Rightarrow Y$) measures the conditional probability of Y given X , i.e., $P(Y|X) = P(XY)/P(X)$.

Additional measures were proposed. Of these, *lift* [51],

$$Lift(X \Rightarrow Y) = P(XY)/P(X)P(Y),$$

is widely used. Lift measures deviations from uniformity and, more specifically, how many times more often X and Y occur together than expected if they were statistically independent. Lift is susceptible to noise in small databases and rare itemsets which occur by chance a few times (or only once) together can produce very large lift values, without any statistical significance.

5.1.2 Dynamic Taxonomy Foci and Association Rules

The market basket model underlying association rules can be mapped into the dynamic taxonomy model by observing that the set I of market-basket items can be

mapped into the set of terminal concepts in dynamic taxonomies,¹ and that the set D of transactions in association rules is equivalent to the set of objects to be classified under concepts in dynamic taxonomies. The dynamic taxonomy model extends (in a manner analogous to [60, 277]) the base association rule model, because it considers a taxonomic organization of items rather than the flat organization assumed by most research on association rules.

As an example, consider a transaction $T = \{\text{beer, chips, light bulbs}\}$. Such a transaction is a dynamic taxonomy object. Each item in T is a terminal concept in the taxonomy. Thus, beer is represented by Beverages > alcoholic > beer, while chips is the terminal concept in Food > snacks > chips. In practice, the translation of a set of market-basket transactions requires: first, that a taxonomy is built from the set of items referenced in each transaction and second, that an object is created for each transaction, and classified under the terminal concepts corresponding to the items referenced in the transaction.

The key concept in dynamic taxonomies is the *reduced taxonomy*, that summarizes the focus set through the original taxonomy, by pruning from the taxonomy those concepts that are not extensionally related to the focus set. Such a pruning can be implemented by eliminating unrelated concepts altogether, or by showing them in an appropriate format while inhibiting their use for further focus refinements.

In producing such a reduced taxonomy for a focus F , $RT(F)$, dynamic taxonomy systems usually count the cardinality of the intersection of each concept C in the taxonomy T with the current focus F , i.e. the related count of C with respect to F : $rc(C|F) = |\text{objects}(C) \cap F|$, for every $C \in T$; $rc(C|U) = |\text{objects}(C)|$, because the intersection of the deep extension of C with the universe U is trivially equal to the deep extension of C .

If we consider the reduced taxonomy $RT(F)$ for the current focus F in terms of association rules, every concept C in the reduced taxonomy $RT(F)$ represents two association rules: $F \Rightarrow C$ and $C \Rightarrow F$, where F is, in general, the conjunct of several concepts, although all boolean operations can be supported.

In this framework, confidence and support can be easily computed as:

$$\text{support}(F \Rightarrow C) = \text{support}(C \Rightarrow F) = P(FC) = rc(C|F)/|U|$$

$$\text{confidence}(F \Rightarrow C) = P(C|F) = P(FC)/P(F) = rc(C|F)/|F|$$

$$\text{confidence}(C \Rightarrow F) = P(F|C) = P(FC)/P(C) = rc(C|F)/rc(C|U)$$

The time overhead for computing these quantities is negligible. As regards storage overhead, we need to store, for each concept $C \in T$, $rc(C|F)$, usually computed anyway, and $rc(C|U)$, i.e. $|\text{objects}(C)|$ in the original taxonomy, which is a constant quantity. In addition, we need the total number of objects in the database, and the cardinality of the current focus.

¹By mapping market-basket items to terminal concepts in DT, the DT terminal level represents “real world” objects.

5.1.3 Integrating Association Rules with Dynamic Taxonomies

Each zoom operation in a dynamic taxonomy updates the current focus F and produces a reduced taxonomy $RT(F)$, which summarizes that focus. By construction, each concept C in $RT(F)$ acts as a representative of two association rules: $F \Rightarrow C$ and $C \Rightarrow F$.

Differently from traditional data mining systems, the discovery of meaningful association rules is interactive. The user can freely focus on the concepts he is interested into, and be presented with “meaningful” association rules that bind the current focus to other concepts. This freedom in exploring association rules solves one of the problems of traditional data mining, namely the fact that many if not most mined rules are not really interesting, so that the identification of interesting rules within a large quantity of noise is often a mining task in itself. In addition, zoom operations can be applied and consequently association rules discovered, at any level of abstraction.

In addition to data mining techniques, human–computer interaction plays a fundamental role. The central problem is how to convey the appropriate information (support, confidence, statistical significance) to the user in a clear and easily perceived way. Although alternate techniques can be used,² we focus on the following graphical representation.

The main indicator is *confidence* vs. *expected confidence (ECF)*. *ECF* is computed assuming a uniform distribution of values and antecedent/consequent independence: $ECF(X \Rightarrow Y) = P(XY)/P(X) = (P(X)P(Y))/P(X) = P(Y)$. Both are represented on the same percentual scale, as shown in Fig. 5.1, where the association rule’s confidence is shown in solid color (green in this case, for reasons discussed below), while the expected confidence is indicated as a vertical white line.

Confidence and expected confidence, represented on the same percentual scale, give an immediate graphical indication of whether the co-occurrence of X and Y is larger or smaller than the one expected under the independence of the two parts of the association rule.

For each of the two rules implied by a concept in the reduced taxonomy, a measure of statistical significance of the deviation of observed from expected confidences is also computed.³ We use the χ^2 test [65], a well known statistical test that is used to evaluate statistically significant differences between proportions for two or more groups in a data set. In this case, we want to determine whether the deviation from independence is significant in the two rules associated to each concept in the reduced taxonomy.

Fig. 5.1 Confidence and expected confidence



²See Chap. 4 on human factors.

³Measures such as lift show the ratio between the two values but do not report the statistical significance of deviation.

Table 5.1 Contingency table for $F \Rightarrow C$

$F \Rightarrow C$	Observed	Estimated
C	$rc(C F)$	$(F/ U) \cdot rc(C U)$
$\sim C$	$U - rc(C F)$	$U - (F/ U) \cdot rc(C U)$

Table 5.2 Contingency table for $C \Rightarrow F$.

$C \Rightarrow F$	Observed	Estimated
F	$rc(C F)$	$(rc(C U)/ U) \cdot F$
$\sim F$	$rc(C U) - rc(C F)$	$rc(C U) - (rc(C U)/ U) \cdot F$

The contingency 1-degree of freedom tables shown in Tables 5.1 and 5.2 are used. The χ^2 value is computed using Yates' correction for continuity [326] in order to avoid the overestimation of statistical significance when frequencies are small, a frequent case after a few zoom operations:

$$\chi^2_{\text{Yates}} = \sum_{i=1}^N \frac{(|O_i - E_i| - 0.5)^2}{E_i}, \quad 0 \leq i \leq 1$$

where O_i are observed values and E_i are estimated values.⁴ The null hypothesis states that the antecedent and the consequent in a rule are independent. It is conservatively rejected at a confidence level of 95% ($\chi^2 \geq 3.841$).⁵

Statistical significance is indicated in the graphical indicators by color-coding in the following way:

- non statistically significant difference: solid gray
- statistically significant difference in which confidence is below expected confidence: solid red
- statistically significant difference in which confidence is above expected confidence: solid green, as shown in Fig. 5.1.

Such a color coding makes the implicit rules $F \Rightarrow \sim C$ and $C \Rightarrow \sim F$ more easily visible.

Finally, we do not use support as normally defined, but for each concept we indicate the ratio $rc(C|F)/|F|$, which gives the support within the focus and hence a picture of how objects in the current focus are subdivided among concepts.

The number of association rules computed during zoom operations can be extended by noting that also the rules $\sim F \Rightarrow C$, $\sim C \Rightarrow F$, $\sim F \Rightarrow \sim C$, and $\sim C \Rightarrow \sim F$ can be computed by similar contingency tables, on the same available data. So, for each concept in the taxonomy, 6 different indicators can be computed. However, the problem of displaying such rules without cluttering the display

⁴In a 1-degree contingency table for an event e , index $i = 0$ records e occurring, and $i = 1$ records e not occurring.

⁵The default confidence level can be varied by knowledgeable users.



Fig. 5.2 The Italian wines infobase

and confusing the user seems quite challenging, even with the use of explanatory tooltips.

5.1.4 An Example

The example presented here is taken from a database tracking Italian wines on sale on wine.com in 2006. These data are taken from a demo application managed by an extended version of Knowledge Processors’ Universal Knowledge Processor [220]. The infobase is online at [221].

Figure 5.2 shows the entire database. As you know, red wines tend to improve with age, while white wines do not. This is confirmed by Fig. 5.3C, where a zoom on red wines is performed. The first bar represents the $F \Rightarrow C$ association rule, while the second bar represents the $C \Rightarrow F$ association rule. As you notice, the second bar in the years 1999 to 2002 indicates that 100% of wines from this vintage are red wines.

Figure 5.4 shows the reduced taxonomy after a zoom on red wines and wines costing more than \$100. Here we notice that there is a significant correlation with the 2001 vintage (an exceptional vintage, actually), and that expensive red wines are significantly from Piedmont. If you just concentrate on 2001, the two bars read respectively:

- More than average “Type: red AND Price: more than 100” wines are “Year: 2001” wines. This is statistically significant

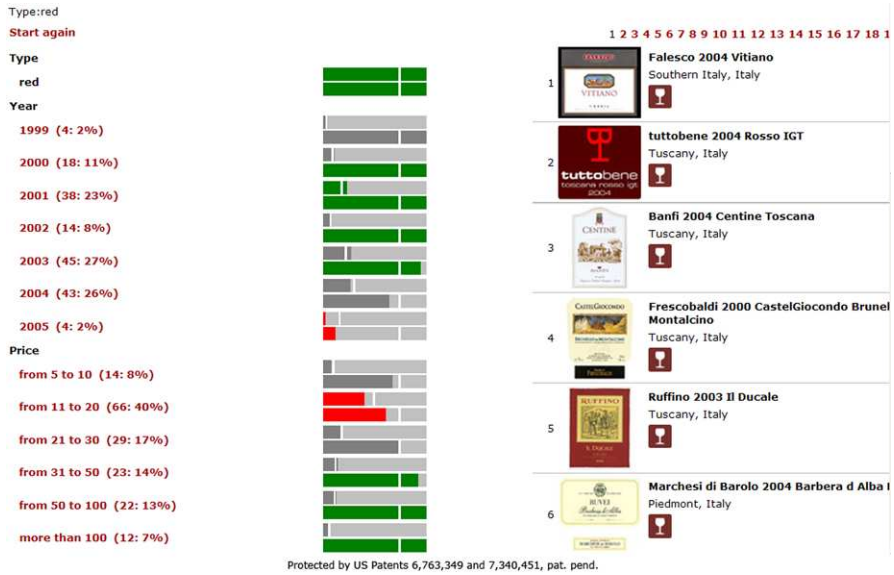


Fig. 5.3 The Italian wines infobase, after a zoom on Red wines

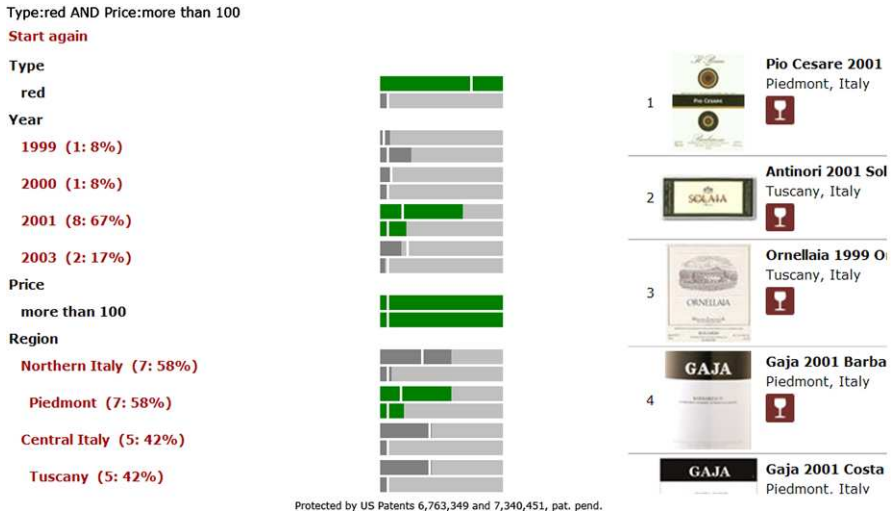


Fig. 5.4 The Italian wines infobase, after a zoom on Red wines and on wines costing more than \$100

- More than average “Year: 2001” wines are “Type: red AND Price: more than 100” wines. This is statistically significant and this is what tooltips on each bar will show to the user.

This example, and especially Fig. 5.4C, shows that it is advisable to avoid pruning unrelated concepts from the reduced taxonomy. Unrelated concepts cannot not be selectable for additional zooms, but the standard measures are as interesting for unrelated as for related concepts, and sometimes even more, especially for wisdom-seeking tasks. In Fig. 5.4, for instance, the negative correlation between expensive red wines and recent vintages is quite important and is not readily seen in the example where unrelated concepts are pruned.

5.1.5 Father–Son Correlation in the Taxonomy

In addition to the association rules discussed above, an indication of whether the documents in the deep extension of concept C are uniformly distributed among C 's sons can be interesting for the user. In the example above, it may be interesting to know if wines are uniformly distributed among years, or some years deviate from uniformity in a significant way. This is a different indicator from association rules because, association rules measure the correlation between the antecedent and the consequent, while here we are concerned with the distribution of data among the sons of the same father in the conceptual hierarchy.

Since these indicators are independent of any antecedent, they can be used at every stage of interaction, including the initial (unreduced) taxonomy. The indicator can be again based on the χ^2 value computed through the contingency table indicated in Table 5.3. For simplicity, we assume that no documents are directly classified under Father.⁶ The extension to non-empty shallow extensions is straightforward.

Deviations from uniformity can be presented through a percentage scale for expected and observed values, as before.

5.1.6 General Association Rules

The associations rules described so far are in the form $F \Rightarrow C$, where F is an expression on concepts in conjunctive normal form, and C is a single concept. In normal

Table 5.3 Contingency table for Father–Son, under a focus $F \subseteq U$, $nSons(Father)$ is the number of Father's sons in F

	Observed	Estimated
<i>Son</i>	$rc(Son F)$	$rc(Father F)/nSons(Father F)$
\sim <i>Son</i>	$rc(Father F) - rc(Son F)$	$rc(Father F) - rc(Father F)/nSons(Father F)$

⁶If they are not, the cardinality of the shallow extension of Father must be subtracted from $rc(Father|F)$ in Table 5.3.

interaction, in fact, each zoom operation ANDs the disjunction of selected concepts with the current focus.⁷

In order to allow for more general association rules in which both the antecedent and the consequent are in conjunctive normal form, we introduce the concept of a *secondary focus*. In normal operations, the focus is iteratively produced by zoom operations starting with an initial focus, which is the universe U . At any point in this zoom sequence, the user can “freeze” the current focus, which becomes the *primary focus* F_1 . Any subsequent zoom operation does not change the primary focus, which becomes a fixed antecedent in the association rule, but defines a secondary focus F_2 , which is applied to the computation of the consequent.

Obviously, the current focus is given by the intersection of the primary and the secondary focus. Equivalently, secondary foci are applied to a universe represented by the primary focus. In this way, conjunctive normal form clauses can be used for both the antecedent and the consequent.

The contingency 1-degree of freedom tables shown in Tables 5.4 and 5.5 are derived from the contingency tables shown in Tables 5.1 and 5.2, by substituting F with F_1 and C with $F_2 \cap C$.

As an example, assume that the current focus is defined as $\{\text{Type: Red} \wedge \text{Year: 2001}\}$. We now freeze this primary focus, which becomes the fixed antecedent for all further analysis until it is “unfrozen”. We now perform a zoom on $\{\text{Price: more than } \$100\}$, which becomes the secondary focus and the prefix of the current consequent. The concept $\text{Location: Northern Italy: Piedmont}$ which survives in the current reduced taxonomy represents part of the consequent in the following association rule:

$$\{\text{Type: Red} \wedge \text{Year: 2001}\} \Rightarrow \{\text{Price: more than } \$100 \wedge \text{Location: Northern Italy: Piedmont}\}$$

Table 5.4 Contingency table for $F_1 \Rightarrow F_2 \cap C$

$F_1 \Rightarrow F_2 \cap C$	Observed	Estimated
$F_2 \cap C$	$rc(F_2 \cap C F_1)$	$(F_1/ U) \cdot rc(F_2 \cap C U)$
$\sim (F_2 \cap C)$	$F_1 - rc(F_2 \cap C F_1)$	$F_1 - (F_1/ U) \cdot rc(F_2 \cap C U)$

Table 5.5 Contingency table for $C \Rightarrow F$

$F_2 \cap C \Rightarrow F_1$	Observed	Estimated
F_1	$rc(F_2 \cap C F_1)$	$(rc(F_2 \cap C U)/ U) \cdot F_1$
$\sim F_1$	$rc(F_2 \cap C U) - rc(F_2 \cap C F_1)$	$rc(F_2 \cap C U) - (rc(F_2 \cap C U)/ U) \cdot F_1$

⁷Conjunct expressions with NOT are also supported.

5.1.7 Side-by-Side Comparison

It is often important to compare two or more views on the same data, but based on different foci. As an example, one might want to compare 2004 wines with 2006 wines, or red wines vs. white wines for all years, or just 2004 red wines with 2006 red wines. Temporal comparison is common, but the examples above show that the temporal facet is just one of the possible foci that can be used.

For simplicity, we assume that only two views are compared. The first view is called the *reference view*, the other view is called the *test view*. While there is only a single reference view, there can be multiple test views.

Initially, the user will manually set the focus for each of the views. Each test view will be compared with the reference view in the following way. The χ^2 test is applied to test whether the distribution of data among concepts in each test view is significantly different from the distribution in the reference view. The test is applied to the same concept C in both views.

In order to allow an unconstrained exploration on data, the user can add one or more concepts to the focus (i.e., zoom) in any view, such concepts being automatically added to the current focus of each view. Thus, after initial different foci are set for all views, all the other zoom operations are “coordinated”, in the sense that the same concept(s) are automatically added to the focus in each view. Such a coordination allows the exploration of different data sets in a meaningful way.

The null hypothesis to be tested is that the distribution of a concept C in the test view is the same as the distribution of C in the reference view. The contingency table to be used is reported in Table 5.6.

The example in Fig. 5.5C shows the comparison of a test view with focus Region: Northern Italy: Piedmont, with a reference view with focus Region: Central Italy: Tuscany. Both views are coordinated on Type: red, i.e., after setting the initial focus for each view, the user performed a zoom on Type: red, coordinating all the other views on this secondary focus. Therefore, the hypothesis to be tested is currently that red wines from Piedmont have the same distribution as red wines from Tuscany. Differences are indicated by arrows, where red arrows⁸ denote statistical significance, and an arrow pointing downwards indicates that the value in the test view is lower than expected. A representation similar to the one shown before could also be used.

Table 5.6 Contingency table for side-by-side comparison

	Observed	Estimated
C	$rc(C F_{\text{view}})$	$(rc(C F_{\text{ref}})/F_{\text{ref}}) \cdot F_{\text{view}}$
$\sim C$	$F_{\text{view}} - rc(C F_{\text{view}})$	$F_{\text{view}} - (rc(C F_{\text{ref}})/F_{\text{ref}}) \cdot F_{\text{view}}$

⁸In gray in the b&w Fig. 5.5.

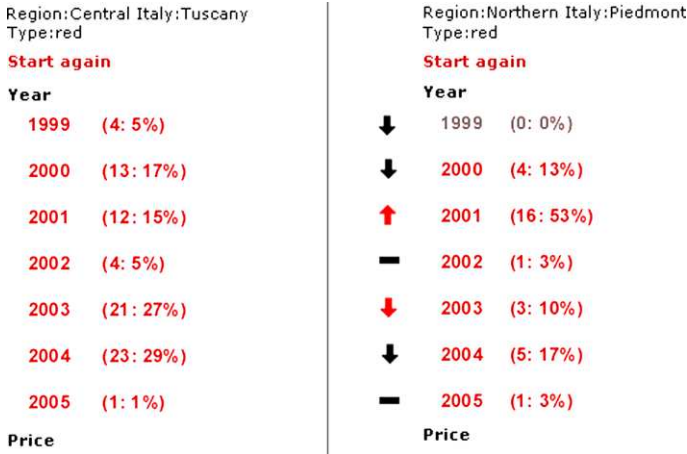


Fig. 5.5 Side-by-side comparison on the Italian wines infobase

5.2 Structured Objects

In the basic dynamic taxonomy model, objects are atomic entities and the extensional inference rule is used to infer unnamed relationships between two concepts if they (or their descendants) are both used to classify at least one object. In some important cases however, objects have an internal structure. Such a structure may be flat as in news summaries (i.e., a single textual document including several different news stories) or hierarchical as in videos, which can be organized on several levels. In these cases, the extensional inference rule can lead to false coordinations or can fail to capture all the interesting relationships.

Consider a news video. It is usually hierarchically organized by news stories, each of which may have several sequences or scenes, which in turn are composed by frames. It seems reasonable to assume that the “strength” of the relationship between two concepts *C* and *C'* decreases for higher levels of this structure. Thus, if two people appear in the same frame they are closely related in space and time, whereas if the same two persons only appear together at the entire video level, their relationship is indirect: it is derived by the fact that they appear in different stories which are part of the same video. Such stories are somehow related, possibly from the temporal point of view (as in a newscast) or because they are about the same topic (as in documentaries).

The meaning of a relationship between two concepts which only occurs at the global level can be difficult to understand by users, and sometimes can lead to false coordinations which make the inference mechanism of dynamic taxonomies seem unreliable. As an example, consider a textual news summary composed by several stories. The first story is about the Prime Minister making a speech in Rome, the second one is about fog in Milan. This object can be classified under terminal concepts such as {political speech, Prime Minister, Rome, fog, Milan}. In this case, a

“false” relationship between fog and Rome, or Prime Minister and Milan will be established by extensional inference.

One might try to solve the problem by “exploding” the news summary into its constituent news stories, and consider each story as an atomic object instead of the entire summary. In this way, the false coordination problem disappears, because fog and Rome will not be related, since they are used to classify different objects.

However, this approach does not really solve the problem for more complex structures, such as news videos. In this case, breaking up the video into its constituent stories, which are considered as objects, only eliminates the indirect relationship inferred at the highest level. However, we still have to account for the fact that two persons appearing in the same frame are more closely (or at least differently) related than two persons appearing in the same news story. In addition, while relationships inferred from the topmost level only do not seem appropriate in a news video, they would be so if we consider a movie. In this case, two actors performing in the same movie seem correctly related even if they do not appear together in any single story.

The approach we propose⁹ [241] explicitly accounts for the structure of objects and allows the user to specify in an intuitive way the appropriate level of granularity for his exploration of the information base.

For simplicity and concreteness, we focus here on video material. In this case, the standard, four-level structure we informally introduced before should capture the vast majority of situations: at the highest level, we have the entire video, which is split into “stories”, each of which is split into “scenes”, each composed of frames. General object structures can conceivably have an arbitrary number of levels, but the approach described here can be easily extended to more general cases.

A multilevel structure description is included in the taxonomy itself by adding a taxonomic facet, “structure”, whose immediate sons are “entire video”, “story”, “scene” and “frame”. Each component (video, story, scene, frame) is an object in itself and is classified under the appropriate concepts. One of these concepts is the appropriate structure type for the object, e.g., scene. Appropriate classification strategies for structured components, a key part of our approach, will be discussed shortly.

First, we note that, because of the PART-OF relationship, each structured level in an object subsumes all the lower levels in the same object. The entire video subsumes all the stories in it, each of which subsumes all the scenes in it, each of which subsumes all the frames in it. If $X \rightleftharpoons Y$ because at least one frame is classified under both concepts, the same relationship holds at the scene level, at the story level, and at the entire video level. This property is similar to subsumption in dynamic taxonomies, and results in a similar *backward inheritance*: each level inherits the classification of its lower levels.

⁹First used in 2001 in a demo system for RAI, the Italian public broadcasting company.

On browsing, the user has two choices. First, he can focus on a specific structure level, say “story”. In this case, only the relationships which are valid at the story level and (because of subsumption) at lower levels, are preserved by the zoom operation: X and Y are related if they appear in (i.e., are used to classify) the same story, or in the same scene or in the same frame. However, X and Y are not related if they only appear at the “entire video” level.

Second, the user can focus on a concept not in the structure facet, e.g., “Sean Connery”. If subsequently he explodes the structure facet, he will see a summary of the structure levels in which Sean Connery appears, so that he can later focus on the appropriate levels.

In order to obtain this type of exploration, component objects of a structured object must be appropriately classified. On classification, the human or automatic classifier will initially structure the document, i.e., tag a sequence of frames as a scene and tag a sequence of scenes as a story. Once the structure is defined, the material has to be classified according to the taxonomy.

Component objects will be classified under the appropriate concepts in the taxonomy. As we mentioned above, a backward inheritance applies. So, if we tag a frame with `Actor > Sean Connery`, Sean Connery is also considered an actor of the scene which subsumes the frame, of the story which subsumes the scene, and of the entire video in which the story appears. Obviously, backward inheritance requires that the union of classifications is taken: thus if Sean Connery is tagged for frame X and Tippi Hedren for frame Y in the same scene S , the actors for S will be {Sean Connery, Tippi Hedren}. The inclusion constraint maintains consistency in the classification, and can be used to describe the document at the lowest, appropriate, structural level: each level “inherits” all the classification descriptors defined at the lower level.

In addition to the “bottom-up” inheritance defined by the inclusion constraint, we can also have a “top-down” inheritance: if the classifier tags the movie with `Director > Alfred Hitchcock` as the movie director, this relationship will hold for all the structural descendants of the movie, down to the frame level.

Consequently, these two types of inheritance can be exploited in order to simplify the classification task. When the user tags a structure element X with a descriptive concept C , C will be added by union to the classification of all the ancestors of X , and C will be added by replacement to all the structural descendants of X .

As an example, consider the movie “Boccaccio 70”, a 1972 Italian movie inspired to the novels by Boccaccio and composed by four stories, each directed by a different director, De Sica, Fellini, Visconti and Monicelli. The classifier will tag each story with his own director. As a consequence, the entire video will be classified under `Director > {De Sica, Fellini, Visconti, Monicelli}`. By specifying the director at the story level, all the lower levels (scenes and frames) will be automatically tagged with the appropriate director.

5.3 Virtual Concepts

Dynamic taxonomies assume that the entire taxonomy is designed and defined before it is used, just as the conceptual schema of a database system. The taxonomy can be obviously modified afterwards, but concepts must be defined before they are used. There are many practical cases in which this requirement can be quite cumbersome, and, most notably, when a subtree (facet) of the taxonomy is used to describe a domain with a potentially very large number of values. Examples include prices, weights, dates, but also authors, performers, etc. Although we can represent them in the current framework, they considerably add to the size and complexity of the taxonomy and might require a careful, anticipatory design. On the other hand, if they are exclusively managed by external search methods, the system will be unable to summarize sets of documents through their facet: in many cases, this will reduce the exploration capabilities of the system.

As an alternative, we propose a new construct, the *simple virtual concept* [231–233], for which neither its actual children (i.e., actual values) nor the actual extension (i.e., the actual objects which have a specific value) are stored, but they are computed from additional, external structures.

A simple virtual concept \bar{V} (e.g., Price) can be fully characterized by four abstract operations:

1. Given \bar{V} , find all its sons. In our example, this means finding all the distinct values of Price in the infobase;
2. Given \bar{V} , find its deep extension: e.g., find all the objects for which Price is defined;
3. Given the son s of \bar{V} , find its deep extension. In the example, find all the objects which have the specified value s of Price; and
4. Given an object d , find all the descendants of \bar{V} under which d is classified. In the example, find all the Prices for a specific object.

Virtual concepts provide a simple way to “virtualize” parts of the taxonomy and materialize such parts, when appropriate, from additional, external structures. Implementation strategies for simple virtual concepts are discussed in Sect. 8.1.7.1.

Simple virtual concepts describe a shallow taxonomic subtree, i.e., a facet and its actual values. Derived virtual concepts allow to group these values in some way which is meaningful for the end-user. A *derived virtual concept* $\delta(\bar{V})$ is derived from a virtual concept \bar{V} by specifying additional restrictions on the virtual concept \bar{V} .

For example, consider the facet “Price” in an e-catalog. With a virtual concept its actual values (i.e., the values of Price used to classify at least one object in the extension) are not explicitly defined in the taxonomy, but are retrieved from external structures through the abstract operations defined above. With a derived virtual concept, we can define an arbitrary hierarchy through additional restrictions. For example, we can specialize “Price” into “Budget”, “Medium”, and “High”, by adding to each abstract operation for the virtual concept representing “Price”, the restrictions “Price < 50” for “Budget”, “Price $\geq 50 \wedge$ Price ≤ 100 ” for “Medium”, “Price > 100” for “High” respectively.

Derived virtual concepts can be derived from other derived virtual concepts, in order to define an arbitrary hierarchy: in this case the additional restrictions are composed in AND. For example, “Bargain” can be derived from the derived virtual concept for “Budget” by adding the restriction “Price < 10”. It is also important to note that derived concepts are dynamic in nature, so that it is conceivable and sometimes beneficial that the end-user be able to dynamically specify custom groupings by entering specific restrictions.

5.4 Logics

In this section, we introduce logics as implicitly-defined taxonomies, where the set of concepts is possibly infinite, and the hierarchy relation is defined as a logical entailment. Classical hand-designed taxonomies can easily be integrated into this framework.

The introduction of logics is motivated by the goal of a tight combination of expressive querying and flexible navigation. Suppose we have documents that are described by a date (e.g., publication date). An expressive query language over these documents should comprise arbitrary date intervals, e.g., “between 10th March 2000 and 14th July 2008”. It is easy to compute the extension of such a date interval given the date associated to each document. In order to provide flexible navigation, a taxonomy should be defined over dates. However, there are too many possible date intervals. If we build a taxonomy out of a small number of date intervals (e.g., years and decades), then we have flexible navigation, but we lose the expressivity as most date intervals cannot be used. Of course, the two systems could be used side by side: the language of date intervals and the taxonomy of some date intervals. However, this could hardly be considered as a *tight* combination; and it would be useful to insert new date intervals from user queries into the taxonomy, for future navigation.

The idea is to define through a logic (syntax and semantics), both an expressive query language and a possibly infinite taxonomy. Then, given a logic and a description of documents, a finite taxonomy is automatically generated as a subset of the logic that is relevant to the descriptions. This taxonomy can be customized by users, in the course of browsing, by adding and removing concepts. In the following, we first define logics as a generalization of taxonomies, then we show how logics can be used in the framework of dynamic taxonomies.

5.4.1 From Taxonomies to Logics

A *taxonomy* can be defined as a partially ordered set of concepts (T, \leq) . These concepts are used in both object descriptions and queries, and the ordering \leq states subsumption relations between concepts. An example is the taxonomy of locations, where concepts are places, regions, countries, continents, etc. Reading a place as “is

somewhere in this place”, the subsumption between places corresponds to spatial inclusion. For instance, $\text{Paris} \leq \text{France}$, but not $\text{Paris} \leq \text{Spain}$; and both $\text{France} \leq \text{Europe}$ and $\text{Spain} \leq \text{Europe}$.

A taxonomy is generally assumed finite, and with an ordering that is explicitly defined, i.e., as an enumeration of concepts and subsumption links. However, it is easy to conceive partial orderings that are both infinite and implicitly defined.

- The set of all strings, ordered by the “contains” relation: "The Jungle Book" \leq "Jungle".
- The set of intervals on integers, ordered by inclusion: $2007 \leq [2000, 2010] \leq [2000, +\infty[$.
- The set of intervals on dates, ordered by inclusion: $3 \text{ sep } 2007 \leq \text{sep } 2007 \leq [\text{sep } 2007, \text{dec } 2007]$.

In these examples the ordering can be given a mathematical definition, and implemented as a function that takes any 2 elements x and y and returns whether $x \leq y$ holds. With these implicitly-defined taxonomies, it is possible to describe a photo with a free comment (string), a size (integer), and a date. Then it is possible to retrieve photos where some word occurs in the comment, that were taken in some period of time, or whose size is in some interval. For navigation, only a finite subset of the concepts are made visible (see Sect. 5.4.2), but the insertion of an additional concept is automatic because its relation to other concepts can be computed.

These implicitly-defined taxonomies can be seen as *logics* because (1) the concepts are properties over objects, and so can be seen as unary predicates, and (2) the ordering is reflexive and transitive, and so can be seen as an entailment. More precisely, this entailment is called *subsumption* (noted \sqsubseteq) because it does not apply to statements, but to properties over objects, as in description logics [90].

Definition 5.1 (Logic) A *logic* is a partially ordered set of formulas (L, \sqsubseteq) , where formulas are used as descriptors on objects and patterns in queries, and the ordering is the *subsumption* relation between formulas, and can be implicitly defined.

A logic is implemented as a module that defines an internal representation, a parser and a printer for user interaction, and a subsumption checker. This definition is compatible with hand-designed taxonomies. The taxonomy of locations is the logic whose formulas are location names, and the subsumption relation is the transitive closure of the taxonomy seen as a graph. In other words, a taxonomy is a particular case of a logic.

Definition 5.2 (Taxonomy as a logic) Let (T, \leq) be a taxonomy. This taxonomy can be defined as a logic (L, \sqsubseteq) by simply stating that $L = T$, and $x \sqsubseteq y$ iff $x \leq y$.

In order to make things more concrete, we now define an example logic that covers common needs for describing objects. We first define the set of formulas, then we define the subsumption ordering over them. A formula is either the

keyword `thing`, which represents the most general formula, or a compound formula $p v$, where p represents a property, and v represents a value for this property. For example, if objects are persons, possible properties are `age`, `birthdate`, `lives_in`, `comes_from`, `position`. Each property takes its values from some domain:

- `age` takes integers and intervals over integers;
- `birthdate` takes dates and intervals over dates;
- `position` takes strings and string patterns (e.g., “contains”, “begins with”);
- `lives_in` and `comes_from` take locations (e.g., towns, countries, continents).

All these formulas are partially ordered by subsumption, from the more general to the more specific. The formula `thing` is the most general, and subsumes all other formulas. It plays the role of the root of a taxonomy. Two compound formulas can be compared if and only if their properties are the same: e.g., one cannot compare an `age` and a `birthdate`. For a given property, formulas are ordered according to their values. A subsumption ordering is defined for each value domain:

- an integer interval subsumes the integers and other intervals it contains (e.g., `in [30, 40]` subsumes `= 32`);
- a date interval subsumes the dates and other intervals it contains, and a low-resolution date subsumes higher-resolution dates (e.g., `2008` subsumes `jul 2008`, which subsumes `13 jul 2008`);
- a string pattern subsumes the strings it matches (e.g., `contains "science"` subsumes `is "computer science"`);
- a location subsumes the locations that are contained in it (e.g., `France` subsumes `Paris`).

The subsumption relation over locations is derived from a hand-designed taxonomy of locations. On the contrary, in other value domains, the subsumption is defined for any 2 values out of an infinite set. Concretely, this means there is an algorithm that decides for any 2 values, whether the first is subsumed by the second or not. Section 5.4.2 shows how dynamic taxonomies can cope with and profit from such infinite partial orderings.

The following arborescence represents a small subset of the logic we just defined. Every item represents a formula, and the child-parent relation represents subsumption.

```

thing
  • lives_in somewhere
    - lives_in Italy
    - lives_in France
      · lives_in Paris
      · lives_in Strasbourg
  • comes_from somewhere
    - comes_from Italy
    - comes_from France
  • birthdate in ..
    - birthdate in [1970,1980]

```

```

· birthdate = 1976
· birthdate = mar 1976
  .birthdate = 19 mar 1976
- birthdate in [1980,1990]
• position contains ""
- position contains "engineer"
- position contains "professor"
· position is "assistant professor in computer science"

```

Section 8.5 presents a framework for allowing application designers to build complex logics from simple and reusable components, without requiring expertise in logic from them.

5.4.2 From Logics to Dynamic Taxonomies

The objective of this section is to show how a logic can fit into the framework of dynamic taxonomies. The principle is that, given a finite subset of formulas X , it is possible to construct a taxonomy from it by defining the set of concepts as X , and ordering them by subsumption, which results in a taxonomy (X, \sqsubseteq) . The important differences with usual taxonomies is that (1) concepts are logical formulas with a syntax and a semantics (not only names), (2) the taxonomy shape is defined through logical inference (subsumption \sqsubseteq), and (3) extensions of concepts are defined through logical inference too, rather than through the taxonomy ordering (see below). The problem of an infinite number of possible concepts is solved by selecting which concepts should appear in taxonomies.

An important issue is which subset X of formulas should be used. The idea is to make it depend on actual data, rather than to define it *a priori*, which would not be very different from designing usual taxonomies. Given a logic (L, \sqsubseteq) , and a description function D that maps each object to a set of formulas in L , we want to specify how the set of concepts X should be derived. Whenever the description function D changes, i.e., addition or removal of an object from the (shallow) extension of a formula, this set X should be updated accordingly. Users should also have the ability to add and remove formulas in the taxonomy in order to customize it.

Suppose we use the logic presented in the previous section, and the following description function:

Object (o)	Properties ($D(o)$)
Martin	birthdate = 12 dec 1961, lives_in Paris, comes_from Strasbourg, position is "engineer"
Luciano	birthdate = 1961, lives_in Strasbourg, comes_from Roma, position is "full professor"
...	...

A first fact is that all formulas appearing in a description should be part of the taxonomy X , because they are obviously relevant to it. However, this is not sufficient because it would make dynamic taxonomies almost flat lists of concepts. In the above example, only the two birthdates would be ordered because we know that Martin and Luciano are born the same year, and we do not know the precise date for Luciano. We can safely exclude from X any formula whose extension is empty as it would never appear in dynamic taxonomies. Of course, when the description function changes, a formula can become relevant by gaining instances, hence the need to synchronize the set X with regard to the description function. A strong advantage of logics is that they enable to define and compute the extension of a formula over a description function, independently of a finite taxonomy of concepts.

Definition 5.3 (Extension) Let (L, \sqsubseteq) be a logic, and D be a description function over a set of objects O . The *extension* of a formula $f \in L$ is defined as the set of objects having a property that is subsumed by f :

$$\text{extension}(f) = \{o \in O \mid \exists d \in D(o) : d \sqsubseteq f\}$$

This is possible because the subsumption is defined (and computable) over an infinite set of formulas, unlike in taxonomies. For instance, in the above example, the extension of `birthdate in [1960, 1970]` contains both Martin and Luciano, and the extension of `comes_from France` contains only Martin.

We may define the set of concepts as the set of all formulas whose extension is not empty,

$$X = \{f \in L \mid \text{extension}(f) \neq \emptyset\}$$

but this is in general too large or even infinite. Indeed, we would get all date intervals containing any date occurring in descriptions, and all substrings of strings occurring in descriptions. An alternative is to extend the logic with an additional operation that returns a set of abstract formulas from a concrete formula.

Definition 5.4 (Abstractions) Let (L, \sqsubseteq) be a logic. An *abstraction operation* over the logic L is a function from any formula $f \in L$ to a finite set of formulas $\text{abstr}(f) \subseteq L$ such that for every $g \in \text{abstr}(f)$ we have $f \sqsubseteq g$.

In the logic defined above, we can define the abstraction operation such that every precise date generates the month, the year, and the decade; every location generates encompassing locations; and, every string generates a substring for each word. For instance, the formula `birthdate = 12 dec 1961` generates the formulas

- `birthdate = dec 1961`,
- `birthdate = 1961`, and
- `birthdate in [1960, 1970]`;

and the formula `position is "full professor"` generates

- position contains "full", and
- position contains "professor".

From this abstraction operation, we can now define the default definition of X , as proposed by the system given a logic and a description function.

Definition 5.5 (Derived taxonomy) Let (L, \sqsubseteq) be a logic, and D be a description function over objects O . The *derived taxonomy* is the partial ordering (X, \sqsubseteq) , where

$$X = \{f \in L \mid \exists o \in O : \exists d \in D(o) : f = d \vee f \in \text{abstr}(d)\}$$

As said above, the derived taxonomy is incrementally synchronized with descriptions. If an object is assigned to a property that is already in X , then only extensions are updated. If an object is assigned to a new property f , then f and all its abstractions are automatically inserted into the derived taxonomy. If, after removing an object from a concept, some concepts have their extension empty, they are removed from the derived taxonomy. In this way, the derived taxonomy is always maintained consistent with the actual data.

Because the derived taxonomy is automatically produced, it may not fit exactly user needs. It may contain spurious concepts, and lack useful concepts. This is why users are allowed to add or remove concepts in the taxonomy at any time, e.g., in the course of browsing the infobase. In order to add concepts, users do not have to place them in the taxonomy. The user just names it by a formula, and the system places it in the taxonomy with the help of the subsumption operation. For instance, a concept for the “summer 2007” can be created by naming the formula `birthdate` in `[21 jun 2007, 22 sep 2007]`. Another way to extend the taxonomy is to query the infobase. All formulas defined in the logic can be used in queries, because their extension can always be computed (see Definition 5.3), and not only formulas in the taxonomy X . Whenever a query entered by a user contains a formula that is not present in the taxonomy and has instances, that formula is inserted into the taxonomy, because if the user has found this formula useful, she may find it useful in the future.

To summarize, logics offer a methodology for automatically generating taxonomies from the descriptions of a collection of objects. Those taxonomies are defined as subsets of possibly infinite partial orderings, whose elements, called formulas, can be complex expressions (e.g., date intervals, string patterns). They are automatically updated upon changes in descriptions, and can also be customized by users. While only a finite and relatively small subset of the logic is available in dynamic taxonomies (for navigation), the full logic is available for querying because the extension can be computed for all formulas, even those absent from the taxonomy. This enables to combine an expressive querying language with the flexible navigation provided by dynamic taxonomies.

5.5 Web Ontologies

The objective of this section is to show how a Web ontology, i.e. a knowledge base as defined in Sect. 3.5.1, can fit into the framework of dynamic taxonomies. The key notions are the classification and instantiation statements that can be inferred from a knowledge base. Classification statements define a subsumption ordering over concepts, which can play the role of a taxonomy. Instantiation statements determine whether an object is an instance of a concept, and hence define the deep extension of concepts. Therefore, all ingredients are present to apply the framework of dynamic taxonomies on web ontologies. The important differences with respect to the usual application of DTs is that (1) the concepts may be complex and in infinite number, and (2) the taxonomy shape and extensions are defined through complex logical inference. The problem of an infinite number of concepts is solved by selecting which concepts should appear in taxonomies, and also by computation-on-demand when expanding concepts (Sect. 5.5.1). The roles that connect objects together seem absent from this picture, but in fact, they appear as parts of complex concepts and are exploited by additional navigation modes (Sect. 5.5.2).

5.5.1 Re-defining Extensions and Dynamic Taxonomies

In dynamic taxonomies, a *local view* is defined at each focus as the combination of a query, an extension, and a dynamic taxonomy. The *query* specifies the current focus, i.e. the user location in the vast navigation space, or her point of view over the whole knowledge base. The *extension* is the set of answers of the query, i.e. a set of objects. The *dynamic taxonomy* gives feedback about this extension, and it is the support of most navigation links. In this section, these three components of local views are formally re-defined on top of description logics (DL).¹⁰ In the following, we assume a DL signature $S = (O, C_a, R_a, Cstr)$, where O is a set of object names, C_a is a set of concept names, R_a is a set of role names, and $Cstr$ is the set of constructors of OWL DL.

Definition 5.6 (Query) A *query* is a (complex) concept, i.e., a combination of object names, concept names, role names, and OWL DL constructors (see Table 3.4). In particular, the three Boolean constructors are available (conjunction, disjunction, negation).

Sometimes, it is useful to see the query as a conjunctive set of simpler concepts, which can be obtained from any query by putting it in conjunctive normal form. In this paper, we use alternately the two forms, whichever is the most convenient. For instance, the query

$$q = Team \sqcap \geq 6 \text{ hasmember.}Person$$

¹⁰See Sect. 3.5.1.

is identical to

$$q = \{Team, \geq 6 \text{ hasmember}.Person\}$$

Next, given a query q , the extension is defined as the set of objects that can be proved to be instances of q .

Definition 5.7 (Extension) Let Σ be a knowledge base and q be a query. The *extension* of q is the set of objects which are instances of this concept:

$$ext(q) = \{o \in O \mid \Sigma \models o : q\}$$

This definition explains why we assume the knowledge base has an A-Box (assertional box). The A-Box introduces objects, assigns concepts to them, and connects them together with roles. Without an A-Box, no instantiation statement could be inferred, and so all extents would be empty. An important property of extensions is that they are monotonic with regard to subsumption. For every concepts C, D , if C is subsumed by D , then the extension of C is included in the extension of D :

$$\Sigma \models C \sqsubseteq D \implies ext(C) \subseteq ext(D)$$

This is consistent with dynamic taxonomies, where we observe the same relation between taxonomic relations and (deep) extensions.

Before defining the dynamic taxonomy associated to a query, we first have to define the taxonomy itself. It is made of a subset of the concept language, ordered by subsumption.

Definition 5.8 (Taxonomy) Let Σ be a knowledge base. The *taxonomy* derived from Σ is the partially ordered set $T_\Sigma = (X_\Sigma, \sqsubseteq_\Sigma)$. The set of concepts X_Σ is the set that contains:

- the most general concept, \top ;
- all concept names in C_a ;
- for every role name $r \in R_a$, every concept name $c \in C_a$, and every natural number $n \in \mathbb{N}$, the concepts $\exists r.\top$, $\exists r.c$, $\geq n r.\top$ and $\geq n r.c$.

Any two concepts $C, D \in X_\Sigma$ are ordered by \sqsubseteq_Σ , i.e. $C \sqsubseteq_\Sigma D$ iff $\Sigma \models C \sqsubseteq D$.

The three Boolean constructors (\sqcap , \sqcup , and \neg) are excluded from the taxonomy because they are easily introduced into queries through the navigation process (see Sect. 4.3). The “at least one of objects” can also be introduced through navigation by a direct selection of objects in the extension. Other constructors are excluded from the taxonomy because they are redundant according to the following equivalences:

- $\forall r.C \equiv \neg \exists r.\neg C$;
- $\leq n r.C \equiv \neg \geq (n+1) r.C$;
- $= n r.C \equiv \geq n r.C \sqcap \leq n r.C$.

Another motivation for not having the constructors \forall , \leq and $=$ is the Open World Assumption (OWA). Indeed, description logics, hence OWL, work under this assumption. For example, in the knowledge base Σ_{ex} of Sect. 3.5.1, we have declared 3 objects as persons and members of the team LIS. The OWA implies that there could be other members of LIS, unknown to the knowledge base designer. Hence, the team LIS is not an instance of the concept $\leq 5 \text{ hasmember.Person}$ (“at most 5 members”), whose extension is empty in our knowledge base. In fact, this extension is empty in most practical knowledge bases, except if a team is explicitly declared to have at most 5 members or less. By excluding these rare concepts from the taxonomy, we make it more compact and more efficient, while retaining the ability to insert them manually in queries.

Now, given a query q , this taxonomy is pruned to retain only concepts that are extensionally related to the query, which results in the *dynamic taxonomy*.

Definition 5.9 (Dynamic taxonomy) Let Σ be a knowledge base, and q be a query. A concept $x \in T_{\Sigma}$ is extensionally related to the query q iff

$$\text{ext}(x) \cap \text{ext}(q) \neq \emptyset$$

This definition is the same as in classical dynamic taxonomies. It can be refined by using a minimal support m (putting $m = 1$ is equivalent to the last definition):

$$\#(\text{ext}(x) \cap \text{ext}(q)) \geq m$$

The *dynamic taxonomy* $DT_{\Sigma}(q)$ is the pruning from the taxonomy T_{Σ} of all the concepts that are not extensionally related to the query q .

Because of concepts $\geq n \text{ r.C}$, there is an infinite number of concepts in the taxonomy T_{Σ} . However, every dynamic taxonomy is finite because a knowledge base is finite, and therefore for every role r and every concept C , there is necessarily a number k such that for every $n \geq k$, the extension of $\geq n \text{ r.C}$ is empty. Still, dynamic taxonomies are often too large to be computed and displayed entirely at once. This is why users are initially presented with a fully collapsed dynamic taxonomy, showing only the most general concept \top , and are allowed to expand concepts on demand, i.e. computing and displaying children concepts. This computation is performed by the function $\text{children}(x, q, m)$ that returns the children of the concept x in $DT_{\Sigma}(q)$, given the current query q and a minimum support m .

We now sketch the definition of the function $\text{children}(x, q, m)$, depending on the shape of the parent concept x :

- $x = \top$: x is the root of the dynamic taxonomy. We must return both concept names and existential restrictions:
 - the most general concept names in C_a ,
 - for every most general role r in R_a (and their inverses), the concept $\exists r.\top$;
- $x \in C_a$: x is a concept name. We must return the most general concept names in C_a that are subsumed by x ;

- $x \equiv \geq n r.C$: x is a qualified cardinality (this includes every concept $\exists r.C$ as an equivalent to $\geq 1 r.C$). We must return:
 - for every most general concept name $D \in C_a$ subsumed by C , the concept $\geq n r.D$,
 - for every most general role name $s \in R_a$ subsumed by r , the concept $\geq n s.C$,
 - the concept $\geq (n + 1) r.C$.

In fact, among the returned concepts, only those that are extensionally related to the query are retained, which requires the computation of an extension and an intersection for each candidate concept. For example, if we apply these definitions to the example knowledge base from Sect. 3.5.1, given a query $q = Person$ and a minimum support $m = 1$, we obtain the following dynamic taxonomy (partial).

⊤

- *Person* (3)
- $\exists \text{memberof}.\top$ (3)
 - $\exists \text{memberof}.\textit{Team}$ (3)
 - * $\exists \text{memberof}.\textit{Bigteam}$ (3)
 - * $\exists \text{leaderof}.\textit{Team}$ (1)
 - $\exists \text{leaderof}.\top$ (1)

More pruning can be done with the help of the T-Box (terminological box). For instance, knowing r is a functional role, it is not necessary to return the increment $\geq 2 r.\top$, because its extension will necessarily be empty. Other prunings can be done with *inverse functional* or *transitive* roles. Also, concepts in the form $\exists r.\top$ could be made more informative by replacing \top by the concept describing the range of the role r (e.g. $\exists \text{memberof}.\textit{Team}$ instead of $\exists \text{memberof}.\top$).

5.5.2 Additional Navigation Modes

In Sect. 4.3, a number of navigation modes are defined. Each navigation mode applies on the current query and a selection to form a navigation link that leads to a new query, hence a new focus. The selection can be a set of concepts from the dynamic taxonomy, or a set of objects. A succession of navigation links leads to queries that are Boolean combinations of concepts. All navigation modes defined in Sect. 4.3 fully apply to web ontologies. However, the existence of objects, roles and quantifiers in description logic concepts necessitates the definition of additional navigation modes to allow their introduction into queries through the navigation process. The following sections show that navigation modes can also apply to parts of queries, and defines three additional navigation modes: *zoom* and *pivot* on object selections, *reversal* on a query part, and *traversal* on a role-based concept.

5.5.2.1 Zoom and Pivot on Object Selection

The navigation modes ‘zoom’ and ‘pivot’ are defined on concepts from the dynamic taxonomy. Because description logics allow to form a concept as an enumeration of objects, it becomes possible to apply the ‘zoom’ and ‘pivot’ on object selections. Given an object selection o_1, \dots, o_n , the selection x to be used by ‘zoom’ or ‘pivot’ is simply defined as the concept $x = \{o_1, \dots, o_n\}$, using the constructor “at least one of objects” (see Table 3.4). For example, starting with the query $\exists \text{ismemberof.Team}$, the extension contains the objects *OLIVIER*, *SEBASTIEN*, and *PIERRE*. Both zoom and pivot on the selection of *SEBASTIEN* and *PIERRE* lead to the new query $\{SEBASTIEN, PIERRE\}$. When selecting a subset of the extension, the zoom has always the same effect as the pivot because of DL inference mechanisms. Every query part subsumes any subset of the extension of the query,

$$\forall q : \forall O \subseteq \text{ext}(q) : \forall y \in q : \Sigma \models O \sqsubseteq y$$

which entails that the whole query is replaced by the object selection. By selecting the negation of an object selection, it is also possible to explicitly exclude some objects from the current focus, like in $\exists \text{ismemberof.Team} \sqcap \neg \{OLIVIER\}$. The use of the constructor “at least one of objects” does not form a new navigation mode, but merely extends zoom and pivot to another type of selection, i.e., sets of objects. This is different from querying by examples (Sect. 4.3.7) because the query is not a combination of concept names derived from the selected objects, but directly uses the explicit name of those objects.

5.5.2.2 Reversal

Roles are already present in dynamic taxonomies through quantified concepts (e.g., $\exists r.C$ and $\geq n r.C$). So, they can be introduced in queries by zoom-in and pivot navigation links. For instance, the following query can be reached in 2 zoom-in steps: $q = \{Bigteam, \exists \text{hasleader.Person}\}$, i.e. “the big teams with a leader”. A useful navigation link would be to reach the related query: $\{Person, \exists \text{isleaderof.Bigteam}\}$, i.e. “the leaders of big teams”, where the role *isleaderof* is defined as the inverse of the role *hasleader*. This navigation mode, called *reversal*, changes the point of view by crossing a role in the query, and turning upside-down the query accordingly. In the above example, the point of view has been changed from teams to persons, through the role *hasleader*.

A query element x can be used as a selection for the reversal mode if it has the form $x = \exists r.C$, and then the new query is defined as:

$$q \leftarrow C \sqcap \exists r^{-1}.(q \setminus \{\exists r.C\}).$$

It can be verified that, if we use again the reversal mode on $\exists r^{-1}.(q \setminus \{\exists r.C\})$, we come back to the initial query q .

5.5.2.3 Traversal

The second type of navigation mode using roles is a composition of two navigation modes defined above, and is called *traversal*. Indeed, it often happens that the user wants to cross a role that is not yet present in the query, but already visible in the dynamic taxonomy. Hence, for a concept $x = \exists r.C$, x is successively used for zoom-in and reversal. These two steps can be simplified by the following definition of the traversal mode:

$$q \leftarrow C \sqcap \exists r^{-1}.q.$$

For instance, we can move in one step from the query *Bigteam* to the query $\{Person, \exists isleaderof.Bigteam\}$ by traversing the concept $\exists hasleader.Person$.

In the definitions of reversal and traversal, we assume that the role r has an inverse that is defined in the knowledge base. If a role has no inverse, we can either prevent it to be reversed or traversed, or we can automatically define an inverse role in the knowledge base.

5.6 Fuzzy Dynamic Taxonomies

Dynamic taxonomies are based on boolean logic and traditional set theory. The probability of the classification of an object o to a concept C , assigned by the classifier, is either 0 or 1. There are many practical situations in which a boolean membership function is not sufficiently accurate and a fuzzy set membership [329] should be used [142, 241]. As an example, a document can have a primary and secondary topics, and each topic used to describe a document should be appropriately weighted, according to its relative importance. Clustering is often fuzzy: for instance, we can assign membership probabilities to the objects in a cluster according to their distance from the centroid of the cluster. As another example, the probability that an actor is associated with a movie can be defined as the ratio between the number of frames in which the actor appears and the total number of frames. This technique allows to model the difference between key and cameo roles. Finally, in medical diagnosis (see Sect. 9.4), the occurrence of symptoms can be more or less frequent in pathologies.

Standard fuzzy set operations are usually defined, after Zadeh [329], as:

- Standard complement: $\sim A(x) = 1 - A(x)$;
- Standard intersection: $(A \cap B)(x) = \min[A(x), B(x)]$;
- Standard union: $(A \cup B)(x) = \max[A(x), B(x)]$

where $A(x)$ is defined as the probability that x belongs to A .

Fuzzy membership has a potential impact on the way selected documents are presented to users: fuzzy document membership can be used to order the list of the documents associated to a concept by decreasing membership probabilities, so that the first documents in the list are those most closely related to the selected concept.

This technique is similar to the *relevance ranking principle* proposed by Robertson and Sparck Jones [229] in the context of text retrieval.

Fuzzy membership also has an impact on the extensional inference rule, used by dynamic taxonomies in order to infer concept relationships. If we account for fuzzy classification, then we must reformulate this rule in order to model fuzzily inferred concepts relationships. In this way, we will have different relationship probabilities among concepts: for instance, two secondary concepts for an object are ‘less’ related than two primary concepts.

The standard fuzzy set operations (and especially fuzzy intersections) can be applied to zoom operations in order to compute fuzzy focuses and fuzzy extensional inference. However, the problem of how to convey fuzzy inference to users in a clear and unambiguous way must be solved. One solution is discussed in Sect. 9.4, where a higher color saturation is used to indicate a higher average membership probability of objects, at any level in the taxonomic tree. Such a coding can effectively guide the user towards the “strongest” relationships and orient thus his search.

A format for exchanging fuzzy descriptions is described in Sect. 8.4.

5.7 Miscellanea

In this section we present a number of useful techniques which, while not extending the model, use some of its features in a non-obvious way.

5.7.1 *Predefined Foci for Personalization and Access Control*

The first technique we describe is the use of a predefined focus. In normal dynamic taxonomy operations, the initial focus is the entire infobase U . There are two situations in which one might want a different initial focus.

First, for personalization. As an example, consider a diagnostic system based on dynamic taxonomies, such as the one described in Sect. 9.4. There, diseases are classified by symptoms and features such as Sex and Age. We can associate a focus to a specific profile, and produce an appropriate reduced taxonomy immediately, instead of the initial taxonomy. As an example, if patient X is male and aged 20, a reduced taxonomy which only retains diseases which can occur to young males can be immediately produced, rather than the more complex initial taxonomy. A number of features are usually recorded in X ’s clinical records so that a fairly accurate profile can be produced and a “minimal” reduced taxonomy produced.

Second, a predefined focus can be used to implement access control for security and privacy reasons. An access profile can be associated to each user as a predefined focus. Such a predefined focus is usually expressed as a combination of concepts, but it can be any subset of the information base. In this way, the user can only see objects he is allowed to see by the predefined focus, and, most importantly, only those concepts which occur in objects he can access. Other concepts will not be shown.

5.7.2 SAES Facets

The dynamic taxonomy model defines a concept as an abstract entity which defines a set of objects. The model does not require a concept to be labeled by a symbol which is necessarily textual. It is interesting to consider concepts which are self-adapting access structures¹¹ themselves, i.e., abstract structures which can be used to query and to summarize results.

Three such SAES are especially interesting in this context: *tag clouds*, *geographic maps* and *object clusters*. We already remarked that tag clouds are indeed SAES if their basic behavior is extended in such a way that on selecting a tag in the cloud, the entire cloud is updated as to represent the set of objects selected. A tag can be seen as concept, and a tag cloud can be seen as a flat dynamic taxonomy. As an alternative, however, we can see the entire tag cloud as an embedded concept which can be used to explore the infobase. In fact, the tag cloud identifies a set of objects and is therefore a concept, according to the definition of concepts in DTs.

Figure 5.6 shows a sample interaction in a e-recruitment application. The user has zoomed on Information technology/Internet as Latest work sector, and now sees the most frequent tags used in the selected curricula. He can then click on any of the tags in the cloud and zoom on it, exactly as if it were any clickable concept in the

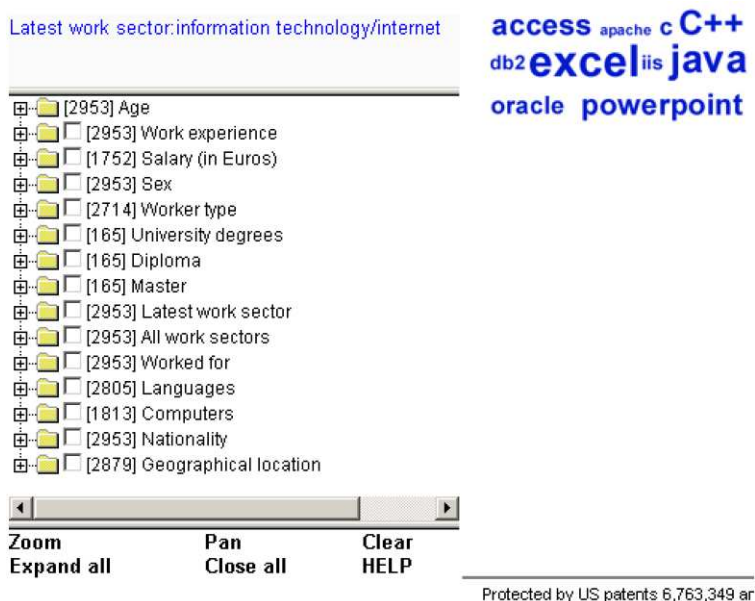


Fig. 5.6 Tag clouds as concepts in a dynamic taxonomy

¹¹See Sect. 2.7.

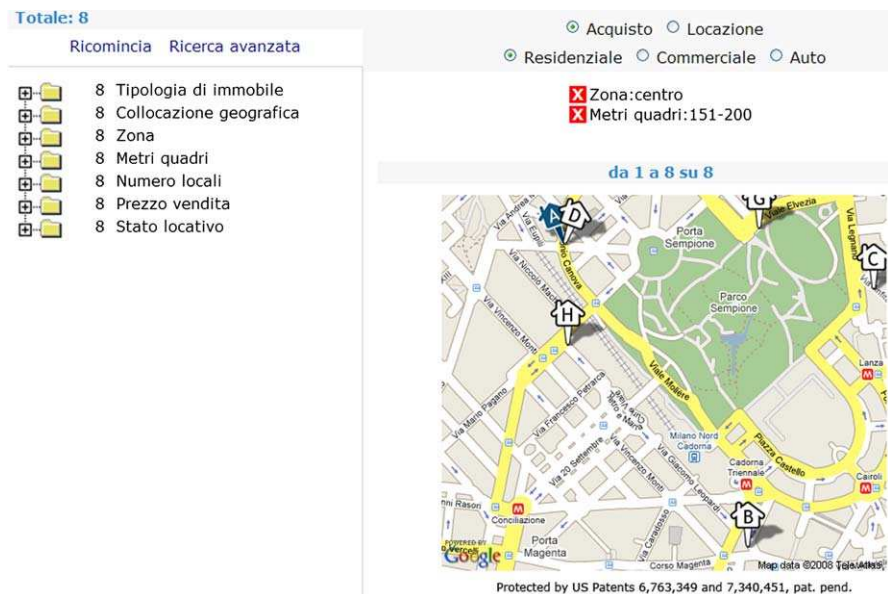


Fig. 5.7 Geographic-enhanced real estate dynamic taxonomy

taxonomy. There can be more than one tag cloud, each with a different semantics: in the example, we could have another tag cloud for spoken languages.

Maps¹² (such as Google Maps) are another type of SAES. A (rectangular) map can be used at the same time for

1. selection, because the geographic coordinates act as a filter, and
2. summary, because objects can be shown on the map itself.

The use of maps integrated in dynamic taxonomies is especially interesting for a number of applications. One of these is real estate portals [233], where the user can query “geographically” and see the features of properties in the selected geographic area or, vice versa, restrict his focus through a combination of non-geographic features, such as price range, number of rooms and see their geographic placement on the map (Fig. 5.7).

Finally, *clustering* (reviewed in Sect. 3.1.2), and especially hierarchical clustering, is again a SAES structure, because it can be used for selection and summaries as well. The justification and use of clustering schemes as facets for accessing image infobases is discussed in details in Sect. 9.3, but this approach carries over to other application domains, such as textual databases.¹³ This solution where clusters represents facets is totally different in practice and in theory from the use of clusters to dynamically describe the result of an information retrieval query (Sect. 3.1.2).

¹²See also Sect. 9.8.

¹³See also Sect. 8.3.1.

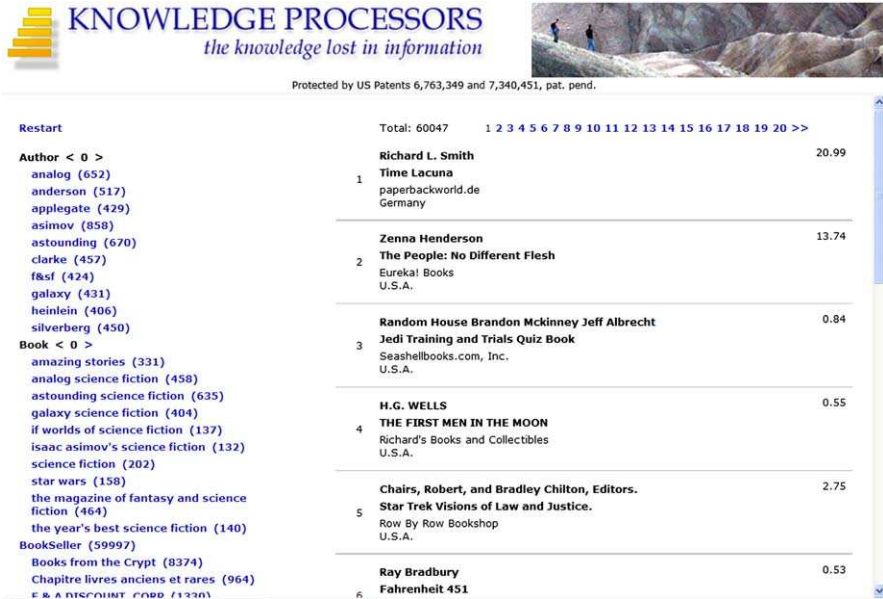


Fig. 5.8 Inventory-based popularity for books in Europe

5.7.3 Popularity, Recommendations, and Authoritativeness

Facets can be used to account for popularity, recommendations and authoritativeness of objects. Popularity is computed as a function of the object, such as number of times the object is accessed, or it is purchased, or simply on the number of copies of objects in the infobase. As an example, Fig. 5.8 shows the popularity of authors (the focus here is Europe) simply on the basis of the number of used books on sale in a global used-book warehouse; Fig. 5.9 shows the global popularity of Asimov's books (not surprisingly *Foundation* is first). Here facet elements are ordered by decreasing number of objects classified under them, in chunks of ten (which keeps the taxonomy readable according to the guidelines of [236]); concepts within each chunk are lexicographically ordered. Interface widgets allow to scroll facet values in chunks of ten at a time. This application is online at [4]. A similar approach is also used in Complete Search DBLP [29].

In many situations, popularity by object counts is not a sufficient indicator. In such cases, a specific facet can account for numeric indicators which can be computed on the basis of number of accesses, number of sales, citations of the object from other objects (especially relevant in scientific literature) or simply on the basis of a grade supplied by users. These important indicators are not generally supported in commercial applications.



Protected by US Patents 6,763,349 and 7,340,451, pat. pend.

Restart Total: 858 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 >>

Author < 0 >
 asimov

Book < 0 >
 earth is room enough (11)
 fantastic voyage (11)
 foundation (21)
 foundation and empire (11)
 foundation's edge (17)
 gold (19)
 i robot (13)
 isaac asimov presents the golden years of science (18)
 nemesis (12)
 robot dreams (11)

BookSeller
 Art Vaughan (17)
 Books from the Crypt (73)
 Chapitre livres anciens et rares (16)
 FAB BOOKS (18)
 Free Shipping Books (16)
 JordanPageBooks (33)
 Mondevana.de (64)
 Powell's Books (23)
 The Book Store (19)
 paperbackworld.de (17)

Nation
 America (622)

	Isaac Asimov			7.56
1	100 Great Science Fiction Short Short Stories		SnowLionBooks U.S.A.	
	Asimov, Isaac			8.22
2	Maverick		E & A DISCOUNT, CORP U.S.A.	
	Asimov, Isaac			8.22
3	Humanidad		E & A DISCOUNT, CORP U.S.A.	
	Asimov, Isaac; Anderson, Paul; Benford, Gregory			1.65
4	Sucker Bait/Un-Man/to the Storming Gulf (Bart Science Fiction Tirplet, No 1)		Centurion Books U.S.A.	
	Isaac Asimov			2.06
5	Science Fiction Die besten Stories von 1943		Upidana Germany	
	Asimov, Isaac			2.21

Fig. 5.9 Inventory-based popularity for Asimov's books

5.7.4 Augmenting IR Recall

Dynamic taxonomies and IR engines¹⁴ are often integrated. The advantages are many: they include smaller taxonomies because rare concepts, and person names can be effectively queried through the IR engine, and an easier user interaction for non-ambiguous terms, which can be entered directly rather than found through the taxonomy.

A well-known problem for IR engines is the semantic gap between the user model (concepts) and the system model (strings). In addition to dealing with inflections, accounting for synonyms and related terms can significantly improve the recall of the user query. Usually, synonyms and terms which are related to query terms are retrieved from a *thesaurus*, such as WordNet [100], and used to modify the user query by *query expansion*.¹⁵

Although a dynamic taxonomy is *not* a thesaurus, it can be modified as to improve query recall through *index expansion* [236] rather than query expansion, in a way similar to a thesaurus.

Concepts in a DT are abstract entities with no relationships with lexical terms. However, we can easily extend the DT framework by allowing the schema designer to specify, for each concept *C*, a set of terms and phrases that are to be added to the

¹⁴See Sect. 3.1.2.

¹⁵See Sect. 3.1.2.

term index for a textual object o , whenever o is classified under C . This approach is complementary to query expansion, in the sense that synonyms and related terms are “added” to the object rather than to the query. Consequently, it can produce larger index structures¹⁶ but does not require thesaurus access at query-time, and it seems more efficient in query-intensive environments.

In addition, index expansion deals easily with implied terms, i.e., terms which do not appear in a textual object because they are implied by its topic. As an example,¹⁷ Leon Battista Alberti’s Renaissance *De Re Aedificatoria* does not contain the term *Renaissance* because it did not exist at the time the book was written. The Unix manual corpus has very few occurrences of the term “Unix”, because it is implied.

Backward inheritance in classification simplifies the selection of terms to be added. Consider the following location index [236]

```
Europe [Europe]
  EU [EU, European Union]
    Italy [Italy]
      Rome
```

where the terms in brackets are index terms to be automatically added when an object is classified under the corresponding concept. The set of index terms added for objects classified under Rome is {Europe, EU, European Union, Italy}. Querying for “Chinese restaurants in Italy” will find the restaurants located in Rome, even if the term Italy does not appear explicitly in the appropriate textual objects.

¹⁶The number of additional terms is usually fairly small.

¹⁷See Sect. 3.1.2.

Chapter 6

Engineering Taxonomy-Based Sources

Yannis Tzitzikas

*“I have made this letter longer than usual,
because I lack the time to make it short.”*

Blaise Pascal, 1623–1662

This chapter is divided into two parts. The first part introduces an algebra that enables the interaction paradigm of dynamic taxonomies even if there are no indexed objects. This algebra is fully *intensional*, in contrast to dynamic taxonomies which are both *intensional* (due to the existence of hierarchies and their semantics) and *extensional* (as they discard concepts with empty extension).

The second part describes other aspects of managing taxonomy-based sources, specifically the adaptation of objects’ descriptions on the basis of user feedback, the creation of inter-taxonomy mappings and issues concerning integration and distributed taxonomy-based sources. Figure 6.1 illustrates some of these tasks.

6.1 Compound Terms Composition Algebra (CTCA)

6.1.1 Motivation

The interaction paradigm of dynamic taxonomies cannot be supported unless we have indexed objects. However it would be useful to have a similar in spirit functionality that could aid the manual indexing (or tagging) of objects, especially in cases where the domain of discourse is wide and there are several facets and terms that are not applicable to every object of the domain. *Compound Term Composition Algebra* (for short *CTCA*) [308] is an algebra that can be used for specifying the meaningful compound terms over a faceted taxonomy in a flexible manner. Below we describe the algebra and several associated issues, i.e. the dynamic generation of navigational trees from CTCA expressions, the revision of CTCA expressions, as faceted taxonomies evolve, the mining of CTCA expressions from materialized faceted taxonomies, and other potential applications of CTCA.

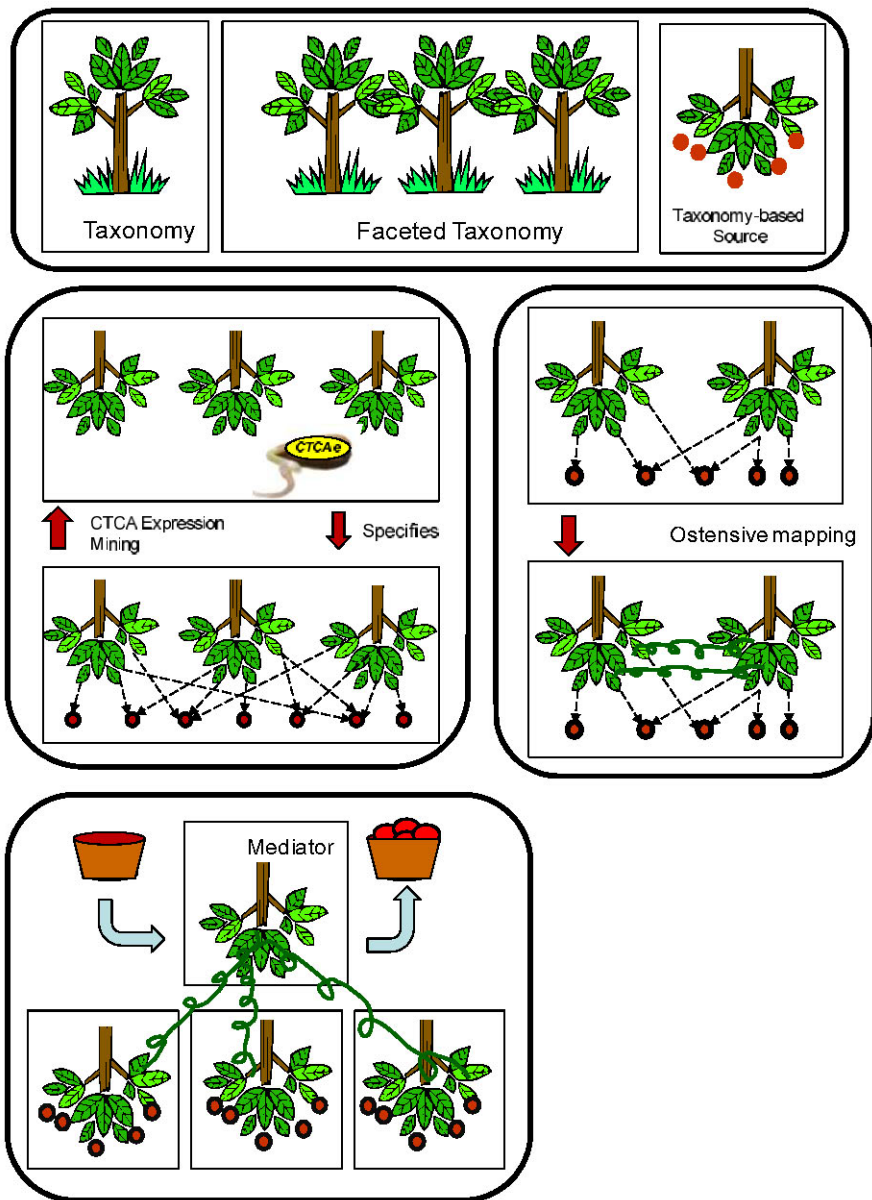


Fig. 6.1 Aspects of Engineering Taxonomy-Based Sources

6.1.2 The Algebra in Brief

As an example, consider the faceted taxonomy of Fig. 6.2 and ignore the indexed objects. One can easily see that several compound terms over this faceted taxon-

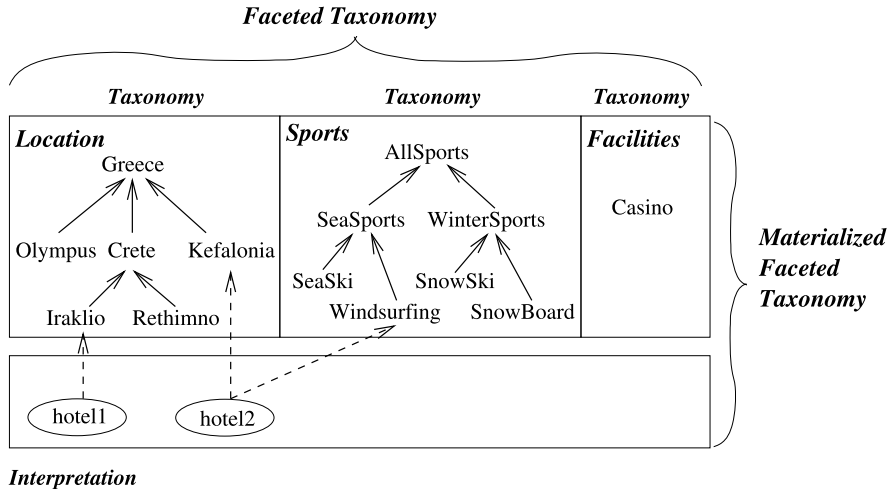


Fig. 6.2 A materialized faceted taxonomy

omy are meaningless, in the sense that they cannot be applied to any object of the domain. For instance, we cannot do any winter sport in the Greek islands (Crete and Kefalonia) as they never have enough snow, and we cannot do any sea sport in Olympus because Olympus is a mountain. For the sake of this example, let us also suppose that only in Kefalonia there exists a hotel that has a casino, and that this hotel also offers sea ski and windsurfing sports.

To specify which compound terms are valid and which are not, one could follow a number of approaches. One approach (which could be called *positive*) would be to construct a list with all valid compound terms (and then assume that the rest are invalid). A second approach (which could be called *negative*) would be to construct a list with all invalid compound terms (and assume that the rest are the valid ones).

A better (less laborious) than the first (positive) approach would be to list only the more specific (narrow) valid compound terms and from these to infer more valid compound terms. The remainder compound terms are considered invalid. For example, if we specify that {Crete, SeaSports} is valid then we can infer that {Greece, SeaSports} is valid too. A better (less laborious) than the second (negative) approach would be to list the more general (broad) invalid compound terms and from these to infer more invalid compound terms. The remainder compound terms are considered valid. For example, if we specify that {Crete, WinterSports} is invalid then we can infer that {Iraklio, WinterSports} is also invalid. CTCA allows following all the above approaches as well as more complex ones that allow combining both valid and invalid compound terms.

CTCA has four basic algebraic operations, namely, *plus-product* (\oplus), *minus-product* (\ominus), *plus-self-product*, ($\overset{*}{\oplus}$), and *minus-self-product* ($\overset{*}{\ominus}$). All these are operations over $\mathcal{P}(T)$, the power set of T , where T is the union of the terminologies of all facets.

A *compound terminology* S is any subset of $\mathcal{P}(T)$ that includes \emptyset . We can order the elements of a compound terminology using the notion of *compound ordering*. A compound term s is *narrower than* (or subsumed by) a compound terms s' , and we write $s \preceq s'$, iff $\forall t' \in s' \exists t \in s$ such that $t \preceq t'$. This ordering is usually referred as Smyth order [273].

For example, and assuming the faceted taxonomy of Fig. 6.2, we have:

$$\begin{aligned} \{Crete, SeaSports\} &\preceq \{Crete\} \\ \{Crete, SeaSports\} &\preceq \{Greece\} \\ \{Crete, SeaSports\} &\preceq \{Greece, SeaSports\} \end{aligned}$$

If s is a compound term we can define:

$$\begin{aligned} BR(s) &= \{s' \in \mathcal{P}(T) \mid s \preceq s'\} \\ NR(s) &= \{s' \in \mathcal{P}(T) \mid s' \preceq s\} \end{aligned}$$

Let S be a compound terminology over T . The broader and the narrower compound terms of S are defined as follows:

$$\begin{aligned} BR(S) &= \{BR(s) \mid s \in S\} \\ NR(S) &= \{NR(s) \mid s \in S\} \end{aligned}$$

The initial operands, thus the building blocks of the algebra, are the *basic compound terminologies*, which are the facet terminologies with the only difference that each term is viewed as a singleton. In most cases, taxonomies are trees, specifically each term has at most one direct parent. The basic compound terminology of such a taxonomy (T_i, \preceq_i) is defined as:

$$S_{T_i} = \{\{t\} \mid t \in T_i\} \cup \{\emptyset\}$$

A definition that captures the general case (i.e. taxonomies containing terms that can have more than one direct fathers) is:

$$S_{T_i} = \{\{t\} \mid t \in T_i\} \cup \{\emptyset\} \cup \{BR(t) \mid t \in T_i\}$$

The motivation for this difference is that every individual term of a taxonomy is by default assumed that it is valid (meaningful), i.e. there are real-world objects (at least one) to which this term applies. It follows that in the taxonomy C of Fig. 6.6, the compound term $\{c2, c3\}$ should be considered valid as it subsumes $\{c4\}$.

An expression e over \mathcal{F} is defined according to the following grammar:

$$e ::= \oplus_P(e, \dots, e) \mid \ominus_N(e, \dots, e) \mid \oplus_P^* T_i \mid \ominus_N^* T_i \mid T_i,$$

where the parameters P and N denote sets of valid and invalid compound terms over the range of the operation, respectively. Roughly, CTCA allows specifying the valid

Table 6.1 The operations of CTCA

Operation	e	S_e
product	$S_1 \oplus \cdots \oplus S_n$	$\{s_1 \cup \cdots \cup s_n \mid s_i \in S_i\}$
plus-product	$\oplus_P(S_1, \dots, S_n)$	$S_1 \cup \cdots \cup S_n \cup BR(P)$
minus-product	$\ominus_N(S_1, \dots, S_n)$	$(S_1 \oplus \cdots \oplus S_n) - NR(N)$
self-product	$\overset{*}{\oplus}(T_i)$	$\mathcal{P}(T_i)$
plus-self-product	$\overset{*}{\oplus}_P(T_i)$	$T_i \cup BR(P)$
minus-self-product	$\overset{*}{\ominus}_N(T_i)$	$\overset{*}{\oplus}(T_i) - NR(N)$

compound terms over a faceted taxonomy by providing a small set of valid (parameter P) and a small set of invalid (parameter N) compound terms. The self-product operations allow specifying the meaningful compound terms over one facet. Specifically, the definition of each operation of CTCA is summarized in Table 6.1, where S_i , $i = 1, \dots, n$, are compound terminologies. Two auxiliary operations, product and self-product are also defined.

If e is an expression, S_e denotes the outcome of this expression and is called the *compound terminology* of e . In addition, (S_e, \preceq) is called the *compound taxonomy* of e .

An expression e is *well formed* iff every facet appears at most once in e , and the parameter sets P and N are always subsets of the corresponding set of *genuine compound terms*. Specifically, each parameter P (resp. N) of an operation $\oplus_P(e_1, \dots, e_k)$ (resp. $\ominus_N(e_1, \dots, e_k)$) should be subset of the set of genuine compound terms over the compound terminologies S_{e_1}, \dots, S_{e_k} , i.e., subset of:

$$G_{S_{e_1}, \dots, S_{e_k}} = S_{e_1} \oplus \cdots \oplus S_{e_k} - \bigcup_{i=1}^k S_{e_i}$$

Returning to the example of Fig. 6.2, the partition of the compound terms to the set of *valid* (meaningful) compound terms and *invalid* (meaningless) compound terms (according to our assumptions), can be defined using the following CTCA expression (in infix notation):

$$e = (\text{Location} \ominus_N \text{Sports}) \oplus_P \text{Facilities}$$

with the following P and N parameters:

$$N = \{\{\text{Crete}, \text{WinterSports}\}, \{\text{Kefalonia}, \text{WinterSports}\}\}$$

$$P = \{\{\text{Kefalonia}, \text{SeaSki}, \text{Casino}\}, \{\text{Kefalonia}, \text{Windsurfing}, \text{Casino}\}\}$$

As another example assume that we want to build a catalog of traditional recipes from all over the world and to this purpose we decide to define facets like *Ingredients*, *LocationOfOrigin* and *CookingStyle* as shown in Fig. 6.3. Notice that several

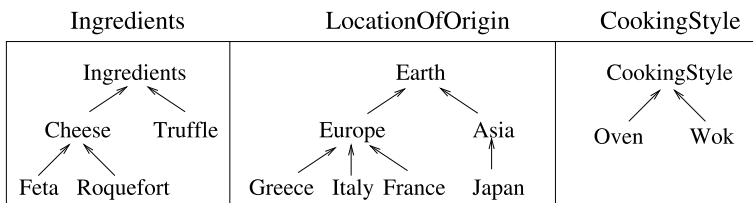


Fig. 6.3 A faceted taxonomy for indexing traditional recipes

combinations of terms are *invalid*, even in this very small domain. For example, the compound term $\{Truffle \text{ (from } Ingredients), Greece \text{ (from } Location)\}$ is invalid as it is impossible to find truffle in Greece, hence there cannot be a traditional Greek recipe that contains truffle. For the same reason the compound term $\{Roquefort \text{ (from } Ingredients), Greece \text{ (from } Location)\}$ is invalid as well as the compound term $\{Feta \text{ (from } Ingredients), France \text{ (from } Location)\}$. Moreover, the compound term $\{Wok \text{ (from } CookingStyle), Europe \text{ (from } Location)\}$ is invalid because wok is used in Asia and not in Europe. According to these assumptions, the partition of compound terms to the set of *valid* (meaningful) compound terms and *invalid* (meaningless) compound terms is shown in Table 6.2. As the facet *Ingredients* has 5 terms, the facet *LocationOfOrigin* has 7 terms, and the facet *CookingStyle* has 3 terms, the number of compound terms that contain at most 1 term from each facet is $6 \cdot 8 \cdot 4 = 192$. This table contains 113 valid and 62 invalid compound terms, thus 175 in total. By adding the $(5 + 7 + 3 = 15)$ singletons (which were omitted from the column of valid) and the empty set we reach the 192 compound terms.

The partition shown in Table 6.2 can be specified using the following very short CTCA expression:

$$e_{recipes} = (Ingredients \oplus_P LocationOfOrigin) \ominus_N CookingStyle$$

with the following P and N parameters:

$$P = \{\{Feta, Greece\}, \{Roquefort, France\}, \{Truffle, France\}, \{Truffle, Italy\}, \\ \{Cheese, Italy\}, \{Cheese, Japan\}\}$$

$$N = \{\{Europe, Wok\}\}$$

Figure 6.4 shows two facets A and B and two expressions: one plus-product and one minus-product each having an empty parameter set. For each case the compound taxonomy that is defined by each expression is shown. Subsequently, Fig. 6.5 shows how the same compound taxonomy, i.e. the same set of compound terms, specifically those enclosed in the continuous curve, can be defined using either a plus-product or a minus-product.

Figure 6.6 shows a faceted taxonomy consisting of three facets, A , B and C . Some examples of compound terminologies that are defined by expressions of CTCA are given in Table 6.3. Note that the empty compound term, i.e. \emptyset , is not

Table 6.2 The valid and invalid compound terms of the Recipe example

Valid		Invalid	
Feta, Gr	Feta, Eu	Feta, It	Feta, Fr
Feta, Ea	Cheese, Gr	Feta, Ja	Feta, Asia
Cheese, Eu	Cheese, Ea	Roquefort, Gr	Roquefort, It
Ingred, Gr	Ingred, Eu	Roquefort, Ja	Roquefort, Asia
Ingred, Ea	Roquefort, Fr	Truffle, Gr	Truffle, Ja
Roquefort, Eu	Roquefort, Ea	Truffle, Asia	Eu, Wok
Cheese, Fr	Ingred, Fr	Gr, Wok	It, Wok
Truffle, Fr	Truffle, Eu	Fr, Wok	Feta, Gr, Wok
Truffle, Ea	Truffle, It	Feta, Eu, Wok	Cheese, Gr, Wok
Cheese, It	Ingred, It	Cheese, Eu, Wok	Ingred, Gr, Wok
Cheese, Ja	Cheese, Asia	Ingred, Eu, Wok	Roquefort, Fr, Wok
Ingred, Ja	Ingred, Asia	Roquefort, Eu, Wok	Cheese, Fr, Wok
Feta, Oven	Feta, Wok	Truffle, Fr, Wok	Truffle, Eu, Wok
Feta, C.Style	Roquefort, Oven	Truffle, It, Wok	Cheese, It, Wok
Roquefort, Wok	Roquefort, C.Style	Ingred, It, Wok	Feta, It, Oven
Cheese, Oven	Cheese, Wok	Feta, It, Wok	Feta, It, C.Style
Cheese, C.Style	Truffle, Oven	Feta, Fr, Oven	Feta, Fr, Wok
Truffle, Wok	Truffle, C.Style	Feta, Fr, C.Style	Feta, Ja, Oven
Ingred, Oven	Ingred, Wok	Feta, Ja, Wok	Feta, Ja, C.Style
Ingred, C.Style	Gr, Oven	Feta, Asia, Oven	Feta, Asia, Wok
Gr, C.Style	It, Oven	Feta, Asia, C.Style	Roquefort, Gr, Oven
It, C.Style	Fr, Oven	Roquefort, Gr, Wok	Roquefort, Gr, C.Style
Fr, C.Style	Eu, Oven	Roquefort, It, Oven	Roquefort, It, Wok
Eu, C.Style	Ea, Oven	Roquefort, It, C.Style	Roquefort, Ja, Oven
Ea, Wok	Ea, C.Style	Roquefort, Ja, Wok	Roquefort, Ja, C.Style
Ja, Oven	Ja, Wok	Roquefort, Asia, Oven	Roquefort, Asia, Wok
Ja, C.Style	Asia, Oven	Roquefort, Asia, C.Style	Truffle, Gr, Oven
Asia, Wok	Asia, C.Style	Truffle, Gr, Wok	Truffle, Gr, C.Style
Feta, Gr, Oven	Feta, Gr, C.Style	Truffle, Ja, Oven	Truffle, Ja, Wok
Feta, Eu, Oven	Feta, Eu, C.Style	Truffle, Ja, C.Style	Truffle, Asia, Oven
Feta, Ea, Oven	Feta, Ea, Wok	Truffle, Asia, Wok	Truffle, Asia, C.Style
Feta, Ea, C.Style	Cheese, Gr, Oven		
Cheese, Gr, C.Style	Cheese, Eu, Oven		
Cheese, Eu, C.Style	Cheese, Ea, Oven		
Cheese, Ea, Wok	Cheese, Ea, C.Style		
Ingred, Gr, Oven	Ingred, Gr, C.Style		
Ingred, Eu, Oven	Ingred, Eu, C.Style		
Ingred, Ea, Oven	Ingred, Ea, Wok		
Ingred, Ea, C.Style	Roquefort, Fr, Oven		
Roquefort, Fr, C.Style	Roquefort, Eu, Oven		
Roquefort, Eu, C.Style	Roquefort, Ea, Oven		
Roquefort, Ea, Wok	Roquefort, Ea, C.Style		
Cheese, Fr, Oven	Cheese, Fr, C.Style		
Ingred, Fr, Oven	Ingred, Fr, C.Style		
Truffle, Fr, Oven	Truffle, Fr, C.Style		
Truffle, Eu, Oven	Truffle, Eu, C.Style		
Truffle, Ea, Oven	Truffle, Ea, Wok		
Truffle, Ea, C.Style	Truffle, It, Oven		
Truffle, It, C.Style	Cheese, It, Oven		
Cheese, It, C.Style	Ingred, It, Oven		
Ingred, It, C.Style	Cheese, Ja, Oven		
Cheese, Ja, Wok	Cheese, Ja, C.Style		
Cheese, Asia, Oven	Cheese, Asia, Wok		
Cheese, Asia, C.Style	Ingred, Ja, Oven		
Ingred, Ja, Wok	Ingred, Ja, C.Style		
Ingred, Asia, Oven	Ingred, Asia, Wok		
Ingred, Asia, C.Style			

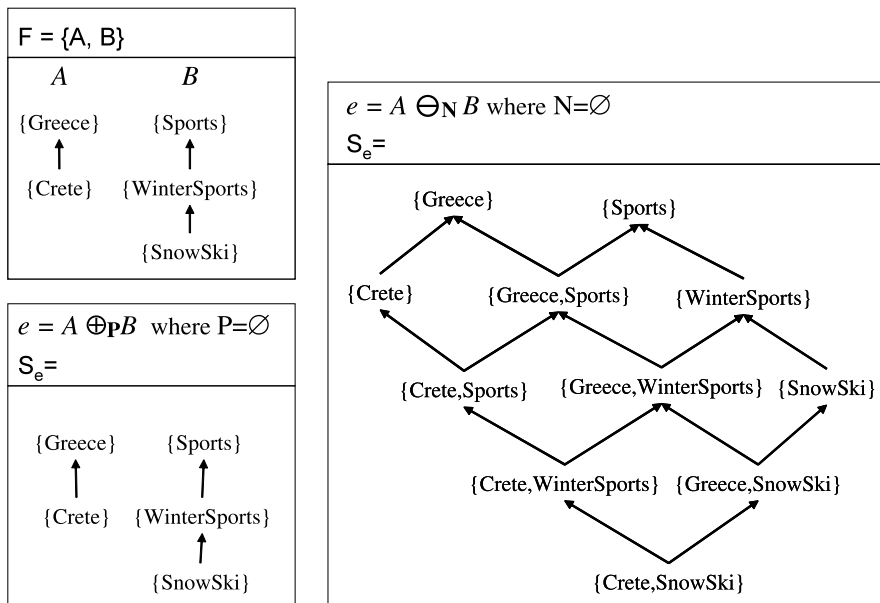


Fig. 6.4 A plus-product and a minus-product both with empty parameters

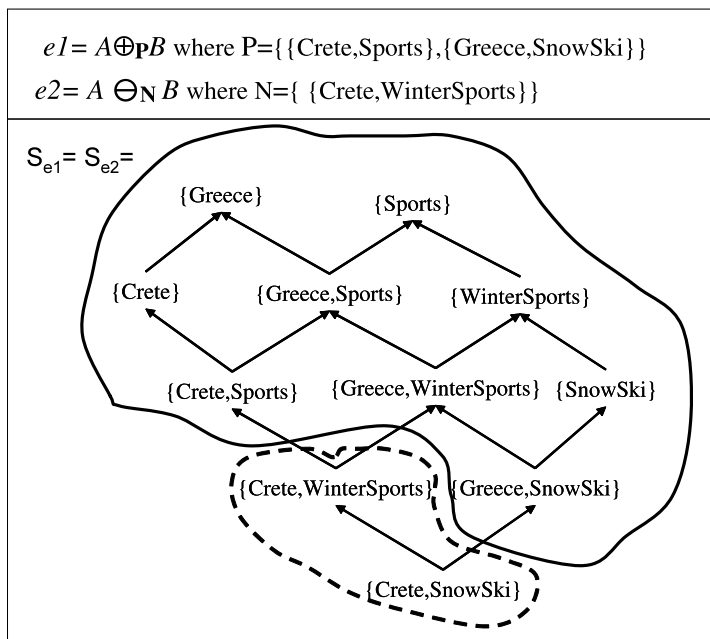


Fig. 6.5 Two equivalent CTCA expressions

Fig. 6.6 A faceted taxonomy consisting of three facets

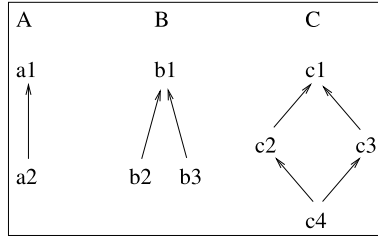


Table 6.3 Some examples of CTCA-defined compound terminologies

e	S_e
$A \oplus_P B, P = \emptyset$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}\}$
$A \ominus_N B, N = \emptyset$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\},$ $\{a1, b1\}, \{a1, b2\}, \{a1, b3\}, \{a2, b1\}, \{a2, b2\}, \{a2, b3\}\}$
$A \oplus_P B, P = \{\{a2, b1\}\}$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}, \{a1, b1\}, \{a2, b1\}\}$
$A \ominus_N B,$ $N = \{\{a1, b2\}, \{a1, b3\}\}$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}, \{a1, b1\}, \{a2, b1\}\}$
$(A \ominus_N B) \oplus_P C,$ $N = \{\{a2, b2\}\},$ $P = \{\{a1, b3, c1\}\}$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}, \{c1\}, \{c2\}, \{c3\}, \{c4\},$ $\{a1, b1\}, \{a1, b2\}, \{a1, b3\}, \{a2, b1\}, \{a2, b3\},$ $\{a1, b3, c1\}, \{a1, b1, c1\}, \{a1, c1\}, \{b3, c1\}, \{b1, c1\}\}$
$(A \oplus_P B) \ominus_N C,$ $P = \{\{a1, b1\}\},$ $N = \{\{b3, c4\}\}$	$\{\{a1\}, \{a2\}, \{b1\}, \{b2\}, \{b3\}, \{a1, b1\}, \{c1\}, \{c2\}, \{c3\}, \{c4\},$ $\{a1, c1\}, \{a1, c2\}, \{a1, c3\}, \{a1, c4\},$ $\{a2, c1\}, \{a2, c2\}, \{a2, c3\}, \{a2, c4\},$ $\{b1, c1\}, \{b1, c2\}, \{b1, c3\}, \{b1, c4\},$ $\{b2, c1\}, \{b2, c2\}, \{b2, c3\}, \{b2, c4\},$ $\{b3, c1\}, \{b3, c2\}, \{b3, c3\},$ $\{a1, b1, c1\}, \{a1, b1, c2\}, \{a1, b1, c3\}, \{a1, b1, c4\}\}$

shown, and for reasons of brevity we adopt the basic compound terminologies for trees, i.e. we do not show the compound term $\{c2, c3\}$ (and all those that contain this compound term).

As we have seen in the previous examples, there are several expressions that could be used for defining the same partition of compound terms. Below we provide some methodological tips that originate from our experience so far.

Consider the compound taxonomies S_1, \dots, S_n and suppose that we have to specify those elements of G_{S_1, \dots, S_n} that are valid. If the majority of the elements of G_{S_1, \dots, S_n} are valid then it is better for the designer to use a *minus-product* operation so as to specify only the invalid compound terms because they are less in number than the set of valid compound terms, so he will have to provide a less number of parameters. Concerning the methodology for defining the set N of a minus-product

operation, it is more efficient for the designer to put in N “short” compound terms that consist of “broad” (w.r.t. \leq) terms. The reason is that from such compound terms a large number of new invalid compound terms can be inferred. In particular, what we are looking for is a Sperner system¹ of the maximal invalid compound terms.

Conversely, if the majority of the genuine compound terms are invalid, then it is better for the designer to employ a *plus-product* operation, so as to specify only the valid compound terms (with a comparatively smaller parameter set). Concerning the methodology for defining the set P of a plus-product operation, it is more efficient for the designer to put in P “long” compound terms that consist of “narrow” terms, since from such compound terms a large number of new valid compound terms can be inferred. In particular, what we are looking for is a Sperner system of the minimal valid compound terms.

Minus-product operations “follow better” the evolution of taxonomies, if they evolve in a top-down manner (addition of leaves). On the other hand, if taxonomies evolve in a bottom-up manner (e.g. assume the case where we have a rather fixed set of leaves and new terms are added on top of these), then plus-product operations follow better evolution.

From an application point of view, it is important to note that there is no need to store the set of valid compound terms that are defined by an expression (i.e. the set S_e), as an algorithm (specifically the algorithm $IsValid(e, s)$ given in [308]) can check whether a compound term s belongs to the set of compound terms defined by an expression e (i.e., whether $s \in S_e$) in polynomial time. Specifically, the computational complexity of this algorithm is $O(|T|^3 \cdot |s| \cdot |\mathbf{P} \cup \mathbf{N}|)$, where \mathbf{P} denotes the union of all P parameters and \mathbf{N} denotes the union of all N parameters appearing in e .² Thus, only the faceted taxonomy \mathcal{F} and the CTCA expression e need to be stored.

6.1.3 Deriving Navigational Trees from CTCA Expressions

As we can infer the valid compound terms of a faceted taxonomy dynamically (through the algorithm $IsValid(e, s)$), we can generate a single hierarchical navigation tree *on the fly*, having nodes that correspond to valid compound terms only. The navigation tree contains nodes that enable the user to start browsing in one facet and then cross to another, and so on, until reaching the desired level of specificity.

The interaction paradigm supported is actually the same with that of dynamic taxonomies. With respect to the formalization described in Sect. 2.6, the only difference is that instead of conditions of the form

$$\bar{I}(ctx) \neq \emptyset$$

¹A *Sperner system* [275] is a set system \mathcal{N} such that if $X, Y \in \mathcal{N}$ and $X \neq Y$ then $X \not\subset Y$ and $Y \not\subset X$.

²Note that $|T|$ is not expected to be very large, in a faceted taxonomy.

here we have conditions of the form

$$ctx \in S_e$$

The former is extensional and intensional (extensional due to I and intensional due to its hat, \bar{I}) while the latter is only intensional.

The algorithm for deriving navigation trees on the fly is described in [308] and has been implemented and used in the FASTAXON system [302] (described in Sect. 8.3.2).

Summarizing, the navigation trees can be exploited for aiding object indexing and preventing indexing errors. For instance, to index (tag) an object the user can browse the navigation tree until forming the desired compound term. The availability of the navigation tree not only could prevent mistakes but it could also be considered as a recommendation service in the sense that during the interaction the user can observe only the applicable facets and those terms of these facets that could be applied. For example, consider a user who wants to index a new recipe. If he has selected the term *Roquefort*, then only *Europe*, *Italy* and *France* will be displayed. from the facet *LocationOfOrigin*.

In addition, browsing the valid compound terms is also useful for the designer of a CTCA expression e for testing whether the compound taxonomy defined by e contains the desired compound terms. This allows following a gradual expression formulation process where the designer can immediately observe the effects of his changes on e .

6.1.4 Tackling the Taxonomy Evolution Problem

Taxonomy updates (additions and deletions of terms or subsumption relationships) may turn a CTCA expression e ill-formed and/or make the compound terminology specified by e no longer reflecting the domain knowledge originally expressed in e .

For instance, consider Fig. 6.3 and $e_{recipes}$. What should we do if we delete the term *Europe* or the term *Asia*, or if we add a new term *Roma* under *Italy*?

A technique that aids the designer to “curate” the expression e would be very helpful and could enhance the robustness of systems that are based on CTCA.

A technique for revising a CTCA expression e after a taxonomy update, so that the new expression e' is well-formed and its semantics (i.e. defined valid compound terms) is as close as possible to the semantics of the original expression e before the update, is possible [294]. Figure 6.7 illustrates the problem. F denotes the original faceted taxonomy, and F' the modified one. S_e^F denotes the compound terminology defined by e , while $S_{e'}^{F'}$ the compound terminology defined by the revised expression e' over the modified F' .

The key idea is to try finding an expression e' such that the distance between $S_{e'}^{F'}$ and S_e^F is minimal. Recall the Principle of Minimal Change [158], which can be found in several forms in the literature, such as the Principle of Persistence of Prior

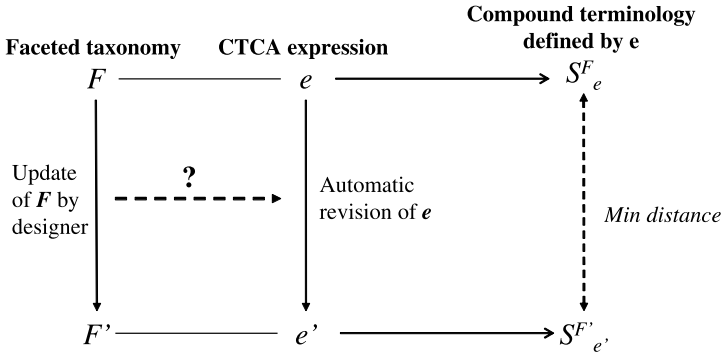


Fig. 6.7 Revising CTCA expressions after taxonomy updates

Knowledge [80], or the Principle of Conservation [115]. This principle states that the new knowledge base should be as close as possible to the original (a thorough discussion can be found at [112]). Of course, closeness or distance has to be defined formally. In our case we define the distance between two compound terminologies S, S' as the cardinality of their symmetric difference (in the classical set-theoretic sense), i.e. we can write:

$$dist(S, S') = |(S - S') \cup (S' - S)| = |S - S'| + |S' - S|$$

A set of basic taxonomy change operations is considered, namely $rename(t, t')$, $delete(t)$, $add(t)$, $delete(t \leq t')$, and $add(t \leq t')$.

Before an operation $delete(t \leq t')$ we assume that the relationship $t \leq t'$ belongs to the transitive reduction (Hasse diagram) of \leq . An example is shown in the left part of Fig. 6.8.

Before an operation $add(t \leq t')$ we assume that the relationship $t \leq t'$ does not already exist in \leq . An example is shown in the left part of Fig. 6.10.

Concerning the deletion of terms we consider that whenever a term t is deleted, all subsumption relationships in which t participates are deleted too. However, as the relation \leq is transitive, if the transitive links of \leq are not stored explicitly, i.e. if only the transitive reduction of \leq is stored, then whenever a term t is deleted, the immediate parent(s) of t should become parent(s) of all immediate children of t . An example is shown in the left part of Fig. 6.9.

Below we describe how e should be revised after each type of taxonomy change operations. In brief, the deletion/addition of terms or subsumption relationships in a faceted taxonomy can be handled by extending the P/N parameters of e , so that the missing compound terms are recovered and the extra ones are removed. Examples are shown in Figs. 6.8 and 6.9. The revision is described in more detail in Table 6.4. The left column lists the taxonomy change operations, except from $add(b \leq a)$ (which we will describe later on), and for each one it shows how e' is derived from e .

Let us first introduce some notations. Given a compound term s and a term t , the notation $s\#t$ is used to denote the compound term $s - \{t\}$. Now given a compound

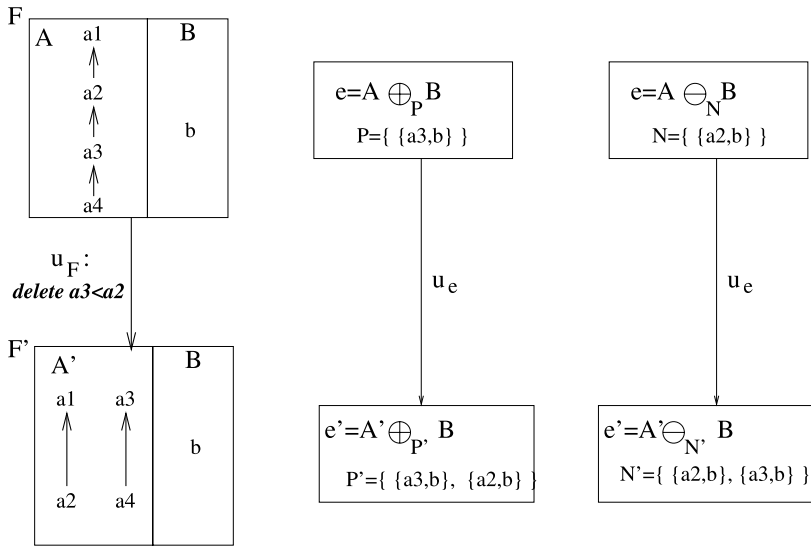


Fig. 6.8 Expression revision after subtraction relationship deletion

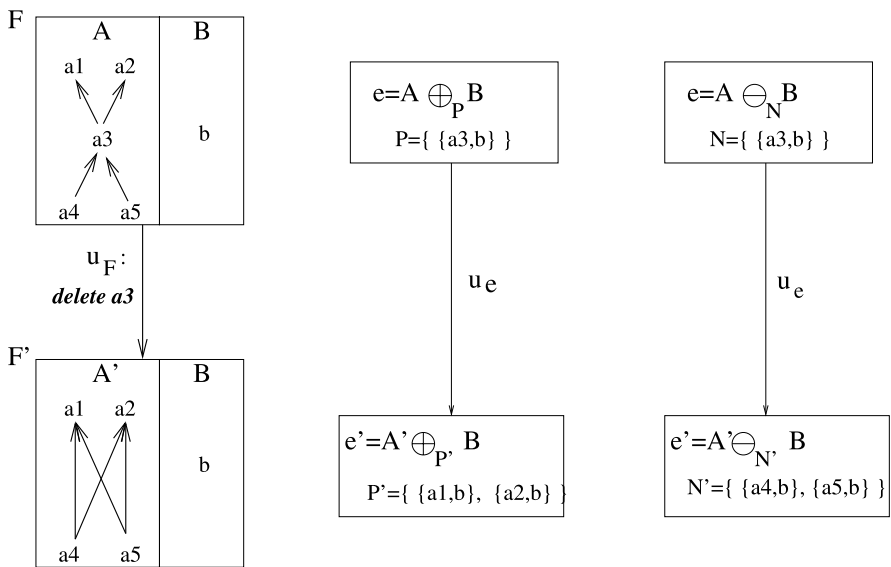


Fig. 6.9 Expression revision after term deletion

Table 6.4 Expression revision after taxonomy updates

	u_F	alg
(1)	rename(a, a')	$P' = \{s\#a\#a' \mid s \in P\}$, $N' = \{s\#a\#a' \mid s \in N\}$ $S_e^{F'} = \{s\#a\#a' \mid s \in S_e^F\}$
(2)	delete(a)	$P' = \bigcup_{s \in P} \{s\#a\#t \mid t \in Br_F^{(1)}(a)\}$ $N' = \bigcup_{s \in N} \{s\#a\#t \mid t \in Nr_F^{(1)}(a)\}$ $S_e^{F'} = S_e^F - \{s \mid a \in s\}$, thus $S_e^{F'} \subseteq S_e^F$
(3)	add(a)	$N' = N \cup \{ \{a, u_i\} \mid e_i \in \text{operands}(e),$ $f(a) \notin f(e_i), u_j \in \text{maximal}_{\leq}(S_{e_i}) \}$ $S_e^{F'} = S_e^F \cup \{ \{a\} \}$
(4)	delete($b \leq a$)	$P' = P \cup \{s\#Nr_F(b)\#a \mid s \in P\}$ $N' = N \cup \{s\#Br_F(a)\#b \mid s \in N\}$ $S_e^{F'} = S_e^F$

term s and two terms t and t' , the notation $s\#t\#t'$ is used to denote the compound term s if $t \notin s$, otherwise the compound term derived from s by replacing t by t' , i.e.:

$$s\#t\#t' = \begin{cases} (s - \{t\}) \cup \{t'\} & \text{if } t \in s, \\ s & \text{otherwise} \end{cases}$$

For example, $\{a, b, c\}\#b\#e = \{a, e, c\}$, while $\{a, b, c\}\#e\#f = \{a, b, c\}$. We can generalize, and for every compound terms $s, s1, s2$, define:

$$s\#s1\#s2 = \begin{cases} (s - s1) \cup s2 & \text{if } s \cap s1 \neq \emptyset, \\ s & \text{otherwise} \end{cases}$$

For example, $\{a, b, c\}\#\{b, c, d\}\#\{e, f, g\} = \{a, e, f, g\}$.

Table 6.4 shows how e' can be obtained for e for each of the basic change operations. In addition it shows the relationship between $S_e^{F'}$ and S_e^F . Specifically, rows (1), (2) and (4) shows how each P and N parameter of e should be revised to a P' and N' parameter of e' . Row (3), which corresponds to term addition, means that for every minus-product operation $\ominus_N(e_1, \dots, e_k)$ and for every e_i ($1 \leq i \leq k$) such that $f(a) \notin f(e_i)$ (meaning the a belongs to a facet that does not appear in expression e_i), we have to add to N the parameter $\{a, u_i\}$ for each $u_i \in \text{maximal}_{\leq}(S_{e_i})$.

However, the addition of a subsumption relationship in a faceted taxonomy cannot be handled, so straightforwardly. The reason is that, since the semantics of the operations \oplus_P/\ominus_N are defined on the basis of the transitive relation \leq (compound ordering), after the addition of a subsumption relationship we may no longer be able to separate (from the semantics) compound terms that were previously separable, i.e., compound terms which were not \leq -related before the addition of the subsumption link. In such cases, the resulting compound terminology of any revised expres-

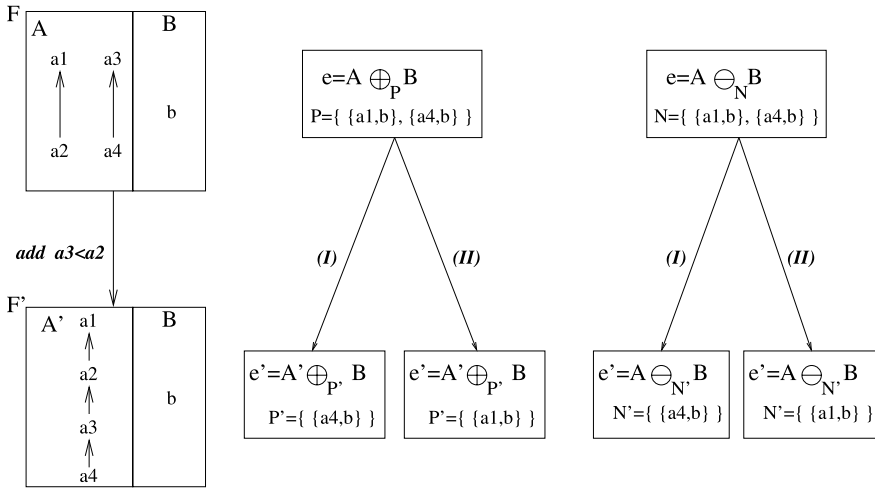


Fig. 6.10 Expression revision after subsumption relationship addition

sion may neither be subset nor superset of the original compound terminology. This happens because the effects of adding a subsumption relationship is different in \oplus_P and \ominus_N : the compound terminologies defined by \oplus_P operations become larger, while those defined by \ominus_N operations become smaller. Now the combination of \oplus_P and \ominus_N operations can lead to compound terminologies which are neither larger nor smaller than the original one. Figure 6.10 shows an indicative example. It shows two expressions and for each one of them two possible revisions. The treatment of such cases as well as additional composite operations (addition of a new terminal term, addition of an intermediate term) are described in detail in [294].

6.1.5 Expression Mining and Other Applications

Assuming a materialized faceted taxonomy M , the problem of automatically deriving an expression e (or the shortest expression e) that specifies all extensionally valid compound terms of M , $V(M)$, is elaborated in [295]. This problem is called *expression mining* and is illustrated in Fig. 6.11.

Let $V(M)$ be the set of all compound terms of a materialized faceted taxonomy M that have a non empty extension, i.e. $V(M) = \{c \in \mathcal{P}(T) \mid \bar{I}(c) \neq \emptyset\}$. Our goal is to find an expression e such that $S_e = V(M)$.

We define the size of an expression e as the number of terms that participate in the P and N parameters of e . The problem of shortest expression mining is to find an expression e such that $S_e = V(M)$ and the size of e is the minimum.

This means that CTCA can be exploited both forthrightly and reversely, i.e., a designer can formulate an expression in order to specify quickly the set of valid compound terms, while from an existing set of valid compound terms an algorithm can find an expression that describes these compound terms. The latter direction

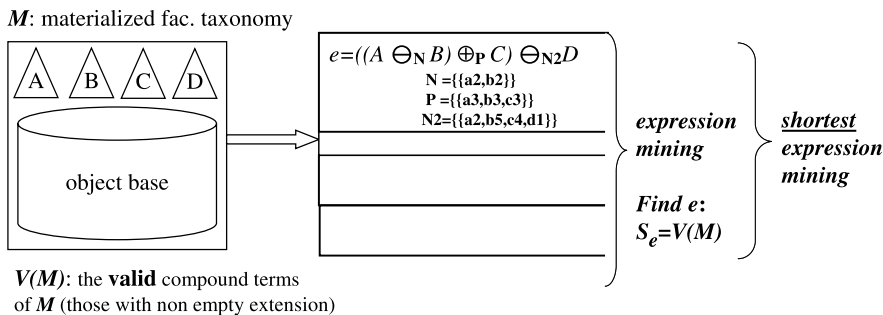


Fig. 6.11 CTCA expression mining

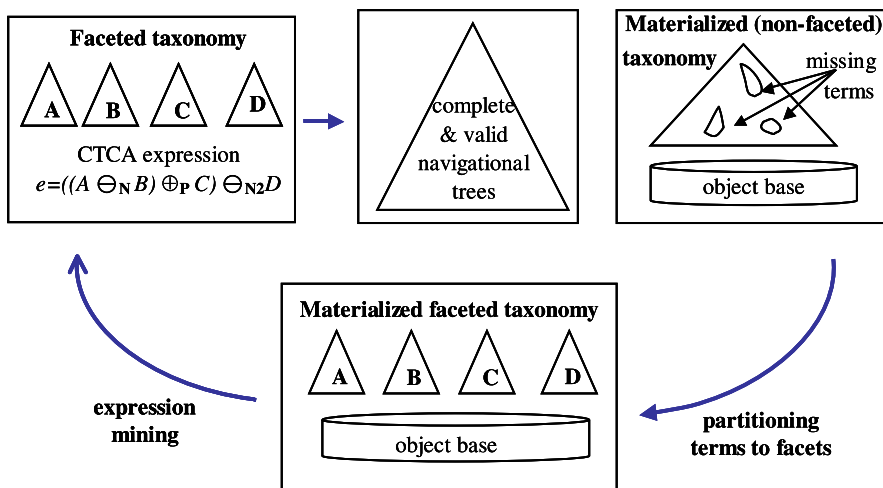


Fig. 6.12 Various scenarios involving CTCA

has several other applications. For instance, it can be used for compressing large symbolic data tables (as shown in [293]), and for exchanging in a compact way the extensionally valid compound terms of a materialized faceted taxonomy.

A complex scenario that involves both directions is illustrated in Fig. 6.12. It sketches a method for reorganizing the single-hierarchical taxonomies on the Web. At first the terms of these taxonomies could be partitioned to facets, resulting to a materialized faceted taxonomy. Subsequently expression mining could be used for expressing all valid combinations as an expression. That expression can then be used for browsing.

Other possible applications of CTCA are described next.

- Speeding up Navigation and Query Services.

The algebra can also be used to speedup navigation and querying. For example, consider the materialized faceted taxonomy of Fig. 6.2, and assume that we have

performed expression mining. Now suppose the user wants to retrieve all hotels located in Greece that offer winter sports. As $\{\text{Crete}, \text{WinterSports}\}$ is an invalid compound term, the system does not have to perform any operation that involves objects.

- Configuration Management

Another application of the algebra is *configuration management*. Consider a product whose configuration is determined by a number of parameters, each associated with a finite number of values. However, some configurations may be unsupported, unviable, or unsafe. For this purpose, the product designer can employ a CTCA expression which specifies all valid configurations, thus ensuring that the user selects only among these.

6.2 Adaptation of Taxonomy-Based Sources Through User Feedback

Dynamic taxonomies and faceted search offer flexible access and interaction. This section describes some basic principles for offering flexibility also in *updates*.

The indexing (description) of objects is inherently imprecise independently of whether it is performed manually or automatically (using statistical methods). One approach that can be used to improve the index, or to adapt it according to the needs and desires of users, is to enable users to provide feedback while interacting with the system. Such feedback can then be exploited to modify the index. In this section we describe index improvement/adaptation/personalization principles and methods. These methods can be specialized according to the particular needs of an application domain. One such domain is that of personal information systems (e.g. for tagging photographs, emails, etc.).

Let S be a function $S : Q \rightarrow \mathcal{P}(Obj)$. Specifically, S is the extension of \bar{I} (for the definition of \bar{I} , see Table 2.1), from T to the set of queries Q . We assume the classical query-and-answer and browsing interaction scheme. In both cases, the “position” of a user while interacting with the system can be described by a pair (q, A) . The query q may have been submitted by the user or it may correspond to his focus while browsing the source through a dynamic taxonomy interface, while A is the answer of that query q , i.e. $A = S(q)$.

Now suppose that the user can modify the current answer A by deleting or adding objects to it (deleting undesired objects, adding desired missing objects), and let A' be the resulting answer. Subsequently the user may request adaptation, i.e. modification of S to an S' such that $S'(q) = A'$.

This means that the user wants a new source S' such that $S'(q) = A'$. There may be several sources S' that satisfy this equation. Here we only consider the case where the solution sources differ only in their interpretation. This means that the taxonomy remains intact and only object descriptions may change. Let I_{sol} be the set of all such interpretations, i.e.

$$I_{sol} = \{I \in \mathcal{I} \mid S_I(q) = A'\}$$

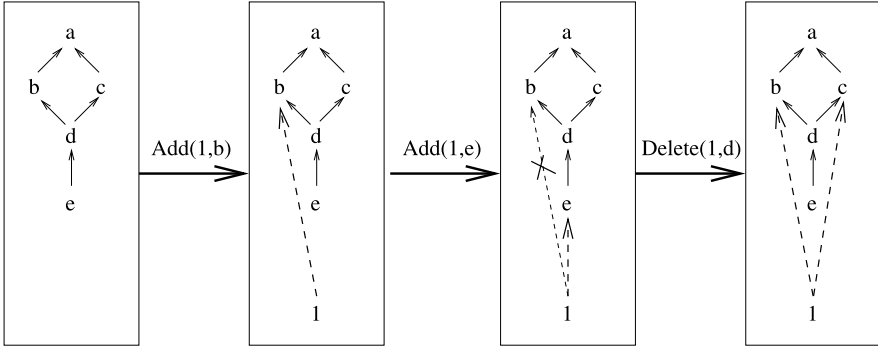


Fig. 6.13 Additions/deletions of single objects from single term queries

where \mathcal{I} denotes the set of all possible interpretations, and $S_I(q)$ denotes $\bar{I}(q)$. We need a criterion for selecting one of them, and it is reasonable to select the interpretation that is closest to the current interpretation I . We can define the distance between two interpretations as follows:

$$dist(I, I') = |\bar{I} \ominus \bar{I}'|$$

where \ominus denote the symmetric difference (in plain set-theoretic sense). We use $\bar{I} \ominus \bar{I}'$ instead of just $I \ominus I'$ in order to take into account the structure of the taxonomy. Now we can define the desired new interpretation I_o as follows:

$$I_o = arg_{I'} minimal \{ dist(I, I') \mid I' \in I_{sol} \}$$

i.e. I_o is the closest (to I) interpretation(s) that can satisfy the user request.³ Technically, we need a method for *modifying* the existing I to the desired I_o rather than generating the entire set I_{sol} and selecting the element that is closest to I . An efficient method for reaching I_o by modifying I is given in [303, 304, 307]. Let $A^+ = A' - A$ and $A^- = A - A'$, so A^+ is the set of objects that have to be added to the current answer, and A^- is the set of objects that have to be deleted from the current answer in order to reach A' , i.e. we can write $A' = (A \cup A^+) - A^-$, or equivalently, $A' = (A - A^-) \cup A^+$. The complexity of the method described in [303, 307] is polynomial. Specifically, the time complexity is

- $O(|A^+| + |A^-| |T|^2)$ for a single term query,
- $O(|A^- \cup A^+| k |T|^2)$ for a disjunctive query of the form $t_1 \vee \dots \vee t_k$, and
- $O(|A^-| k |T|^2 + |A^+| k)$ for a conjunctive query of the form $t_1 \wedge \dots \wedge t_k$.

Some indicative simple examples are shown in Fig. 6.13. It shows the results of requests for adding/deleting an object to/from the interpretation of a single term.

³There can be more than one such interpretations.

It is not hard to see that the same machinery could be used for other kinds of user feedback. For instance, instead of modifying the answer set the user can change the query, i.e. from q he may derive a q' , and request from the system to adapt, i.e. to reach a S' such that $S'(q') = A$. We can exploit the same technique by considering $A^+ = A - S(q')$ and $A^- = S(q') - A$.

This section presented only the foundations of this method. In real applications this method can be extended with various other aspects, e.g. for specifying which facets are updatable and which are not, for specifying criteria (or ranking methods) for breaking ties (when there are several interpretations I_o that have minimal distance from I), etc.

6.3 Mapping Taxonomy-Based Sources

Assume that we have two taxonomy-based sources and suppose that we want to establish mappings between their taxonomies. If these sources share instances, i.e. if there are objects classified with respect to both taxonomies, then the *ostensive* method [296] can be used. The ostensive method is a data-driven method for establishing mappings. Cornerstone of the ostensive method is what is called *naming function* (also analyzed in [307]). Intuitively, a naming function is a function that takes as input a set of objects A and returns a query whose answer is the set A or a set that is as “close” as possible (in set-theoretic sense) to A .

Let S be a function $S : Q \rightarrow \mathcal{P}(Obj)$. Specifically, S is the extension of \bar{I} (for the definition of \bar{I} , see Table 2.1), from T to the set of queries Q .

Ideally we would like a function $n : \mathcal{P}(Obj) \rightarrow Q$ such that:

$$\forall A \subseteq Obj, \quad S(n(A)) = A$$

If such a function exists we call it *exact naming* function. If $S(n(A)) = A$ we say that $n(A)$ is the *exact name* of A .

However as S is not always a surjective function we introduce “approximate” naming functions. Let A be a set of objects ($A \subseteq Obj$). We define a *lower naming* function n^- and an *upper naming* function n^+ , as follows:

$$n^-(A) = \text{lub}_{\leq} \{q \in Q \mid S(q) \subseteq A\}$$

$$n^+(A) = \text{glb}_{\leq} \{q \in Q \mid S(q) \supseteq A\}$$

where *lub* stands for least upper bound, and *glb* stands for greatest lower bound with respect to the query containment ordering. Query containment is defined as follows: given two queries, q and q' in Q , we write $q \leq q'$ if $S(q) \subseteq S(q')$, and we write $q \sim q'$, if both $q \leq q'$ and $q' \leq q$ hold. Note that \sim is an equivalence relation over Q , and let Q_{\sim} denote the set of equivalence classes induced by \sim over Q . Note that \leq is a partial order over Q_{\sim} .

Now let A be a subset of Obj for which both $n^-(A)$ and $n^+(A)$ are defined, i.e. the above *lub* and *glb* exist (each corresponding to one element of Q_{\sim}). It is clear

that in this case it holds:

$$S(n^-(A)) \subseteq A \subseteq S(n^+(A))$$

and that $n^-(A)$ and $n^+(A)$ are the best “approximations” of the exact name of A . Note that if $S(n^-(A)) = S(n^+(A))$ then both $n^-(A)$ and $n^+(A)$ are exact names of A .

Let Q_N denote the subset of Q that contains queries that we would like to consider as candidate names. In general, $Q_N = Q$. However we introduce Q_N because we may want names to be queries of a specific form. For instance, we may want to exclude queries that contain negation.

If Q_N is a query language that (a) supports disjunction (\vee) and conjunction (\wedge) and is closed with respect to these, and (b) has a top (\top) and a bottom (\perp) element such that $S(\top) = Obj$ and $S(\perp) = \emptyset$, then the functions n^- and n^+ are defined for every subset A of Obj . In this case (Q_{\sim}, \leq) is a complete lattice and these functions are defined as:

$$n^-(A) = \bigvee \{q \in Q_N \mid S(q) \subseteq A\}$$

$$n^+(A) = \bigwedge \{q \in Q_N \mid S(q) \supseteq A\}$$

As Q_N is usually an infinite language, $n^-(R)$ and $n^+(R)$, as defined earlier, are queries of infinite length. This means that in practice we also need a method for computing a query of finite length that is equivalent to $n^-(R)$ and another one that is equivalent to $n^+(R)$.

Given an interpretation I and an object o , let $D_I(o) = \bigwedge \{t \in T \mid o \in I(t)\}$. Let $Q_N = Q^+$ where Q^+ denotes the set of boolean expressions of terms that do not contain negation. It can be proven that:

$$n^+(A) \sim \bigvee \{D_I(o) \mid o \in A\}$$

$$n^-(A) \sim \bigvee \{D_I(o) \mid o \in A, S(D_I(o)) \subseteq A\},$$

The time complexity for computing these names is polynomial. Specifically, the time complexity is $O(|A||T|)$ for computing $n^+(A)$ and $O(|A||Obj||T|^2)$ for computing $n^-(A)$.

The naming functions now will be exploited for establishing mappings. Consider two sources $S_i : Q_i \rightarrow \mathcal{P}(Obj_i)$, and $S_j : Q_j \rightarrow \mathcal{P}(Obj_j)$. Ostensive mapping is possible only if their domains are not disjoint, i.e. if $Obj_i \cap Obj_j \neq \emptyset$. Let C denote their common domain, i.e. $C = Obj_i \cap Obj_j$. The ostensive method can yield relationships that are extensionally valid in C .

Suppose that S_i wants to establish a mapping $m_{i,j}$ to a source S_j . A mapping $m_{i,j}$ can contain relationships of the form:

- (i) $q_i \geq q_j$,
- (ii) $q_i \leq q_j$

where $q_i \in Q_i, q_j \in Q_j$. These relationships have the following meaning:

- (i) $q_i \geq q_j$ means that $S_i(q_i) \cap C \supseteq S_j(q_j) \cap C$
- (ii) $q_i \leq q_j$ means that $S_i(q_i) \cap C \subseteq S_j(q_j) \cap C$

The form (i) or (ii) of the relationships of a mapping depends on the intended use of the mappings. For instance, suppose that S_i acts as a *mediator* [316] over S_j . If S_i wants to compute *sound* (with respect to C) answers, then it should use only relationships of type (i) during query translation. On the other hand, if S_i wants to compute *complete* (with respect to C) answers then it should use relationships of type (ii) (e.g. see [165]).

Another remark is that if S_i is a mediator that adopts a *global-as-view*⁴ modeling approach, then all q_i that appear in $m_{i,j}$ are primitive concepts. On the other hand, if S_i adopts a *local-as-view* approach then all q_j that appear in $m_{i,j}$ are primitive concepts of S_j .

Ostensive mapping for the more general case where S_i is interested in relationships of both, (i) and (ii), types, and where q_i, q_j can be arbitrary queries, is described next. Let n_j^- and n_j^+ be the naming functions of S_j as defined earlier. Also let $S_i^c(q) = S_i(q) \cap C$ and $S_j^c(q) = S_j(q) \cap C$. Now suppose that S_i wants to articulate a query $q_i \in Q_i$. The query q_i should be articulated as follows:

- $q_i \geq n_j^-(S_i^c(q_i))$ if $S_i^c(q_i) \supseteq S_j^c(n_j^-(S_i^c(q_i)))$
- $q_i \leq n_j^-(S_i^c(q_i))$ if $S_i^c(q_i) \subseteq S_j^c(n_j^-(S_i^c(q_i)))$
- $q_i \geq n_j^+(S_i^c(q_i))$ if $S_i^c(q_i) \supseteq S_j^c(n_j^+(S_i^c(q_i)))$
- $q_i \leq n_j^+(S_i^c(q_i))$ if $S_i^c(q_i) \subseteq S_j^c(n_j^+(S_i^c(q_i)))$

Observe the role of the naming functions. S_j instead of checking all queries in Q_j , it just uses its naming functions in order to compute the lower and the upper name of the set $S_i(q_i) \cap C$. Recall that the naming functions (by definition) return the most precise (semantically close) mapping for q_i .

It is worth mentioning that the above relationships can be obtained without extensive communication. In fact, they can be obtained by a quite simple and efficient (in terms of exchanged messages) distributed protocol. Fig. 6.14 illustrates the protocol (i.e., the exchanged messages) for linking a query q expressed over the terminology of S_1 with a query expressed over the terminology of S_2 . Each source has to send only one message.

The exact protocol is described in Fig. 6.15. Two messages have to be exchanged between S_i and S_j for articulating the query q_i .

A source can run the above protocol in order to articulate one, several or all of its terms (or queries).

Another remark is that S_i and S_j need not a-priori know (or compute) their common domain C , as C is “discovered” during the execution of the protocol (this is the reason why S_j stores in F and sends to S_i those objects that do not belong to Obj_j).

⁴For more about the distinction *global-as-view* versus *local-as-view* see [56, 165].

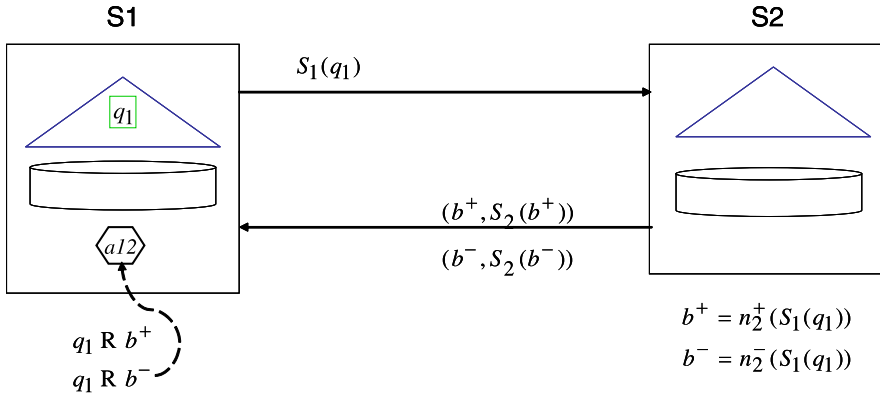


Fig. 6.14 Messages for articulating q_1 of S_1 according to the ostensive method

- S_i : (1) $A := S_i(q_i)$;
 (2) $\text{SEND}_{S_i \rightarrow S_j}(A)$
- S_j : (3) $F := A \setminus \text{Obj}_j$
 (4) $A := A \cap \text{Obj}_j$;
 (5) $\text{down} := n_j^-(A)$; $B\text{down} := S_j(\text{down})$;
 (6) $\text{up} := n_j^+(A)$; $B\text{up} := S_j(\text{up})$;
 (7) $\text{SEND}_{S_j \rightarrow S_i}(F, \text{down}, B\text{down}, \text{up}, B\text{up})$
- S_i : (8) If $(A \setminus F) \supseteq (B\text{down} \cap \text{Obj}_i)$ then set $q_i \geq \text{down}$;
 (9) If $(A \setminus F) \subseteq (B\text{down} \cap \text{Obj}_i)$ then set $q_i \leq \text{down}$;
 (10) If $(A \setminus F) \supseteq (B\text{up} \cap \text{Obj}_i)$ then set $q_i \geq \text{up}$;
 (11) If $(A \setminus F) \subseteq (B\text{up} \cap \text{Obj}_i)$ then set $q_i \leq \text{up}$

Fig. 6.15 The ostensive articulation protocol

If the desired query language \mathcal{Q}_N does not satisfy the criteria mentioned earlier (i.e. lub and glb do not always exist), then the only difference of the protocol is that the message that S_j sends to S_i may contain more than one up and $down$ queries. This is clarified later on by examples.

Consider the sources shown in Fig. 6.16 and suppose that S_1 wants to articulate its terms with queries of S_2 . In the following examples we omit the set F (from the message of line (7) of Fig. 6.15) as it is always empty.

The steps for articulating the term *cabbages* follow:

- $S_1 \rightarrow S_2$: {1}
 $S_2 \rightarrow S_1$: (\perp, \emptyset), (green, {1, 5, 6})
 S_1 : cabbages \leq green

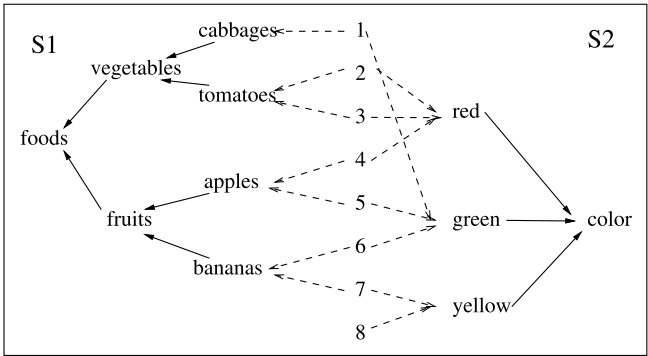


Fig. 6.16 An example of two sources S_1 and S_2 . Here $Obj_1 \cap Obj_2 = \{1, 2, 3, 4, 5, 6, 7\}$

The steps for articulating the term *apples* follow:

- $S_1 \rightarrow S_2$: $\{4, 5\}$
- $S_2 \rightarrow S_1$: $(\perp, \emptyset), (red \vee green, \{1, 2, 3, 4, 5, 6\})$
- S_1 : $apples \leq red \vee green$

The steps for articulating the term *foods* follow:

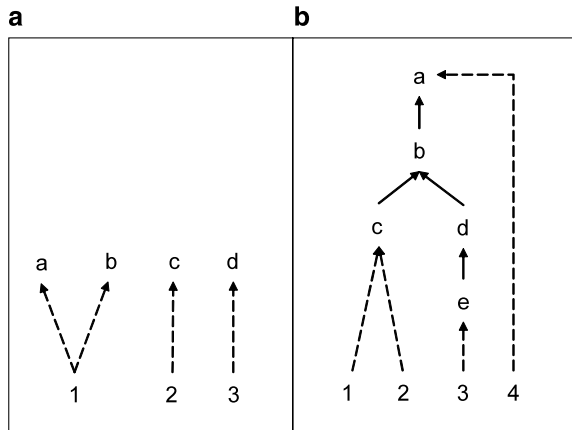
- $S_1 \rightarrow S_2$: $\{1, 2, 3, 4, 5, 6, 7\}$
- $S_2 \rightarrow S_1$: $(red \vee green, \{1, 2, 3, 4, 5, 6\}),$
 $(red \vee green \vee yellow, \{1, 2, 3, 4, 5, 6, 7, 8\})$
- S_1 : $foods \leq red \vee green,$
 $foods \sim red \vee green \vee yellow$

If S_1 runs the protocol for each term of its taxonomy, it will infer the following relationships:

- cabbages \leq green
- tomatoes \leq red
- apples \leq red \vee green
- bananas \leq green \vee yellow
- vegetables \leq green \vee red
- fruits \leq red \vee green \vee yellow
- foods \leq red \vee green
- foods \sim red \vee green \vee yellow

If S_2 runs this protocol for each term of its taxonomy, it will infer the following relationships:

Fig. 6.17 An example of two sources



$red \geq tomatoes$
 $red \leq tomatoes \vee apples$
 $green \geq cabbages$
 $green \leq cabbages \vee apples \vee bananas$
 $yellow \leq bananas$
 $color \sim cabbages \vee tomatoes \vee apples \vee bananas$

The protocol can be used to articulate single terms to queries, but also to articulate queries to queries. For example, the steps for articulating the query $apples \vee bananas$ follow:

$S_1 \rightarrow S_2 : \{4, 5, 6, 7\}$
 $S_2 \rightarrow S_1 : (red \vee green \vee yellow, \{1, 2, 3, 4, 5, 6, 7, 8\})$
 $S_1 : apples \vee bananas \leq red \vee green \vee yellow$

Now consider the case where we do not want to articulate terms with queries, but terms with *single terms* only, i.e. consider the case where $Q_N = T$. Note that now $lub\{t \in T \mid S(t) \subseteq A\}$ and $glb\{t \in T \mid S(t) \supseteq A\}$ do not always exist. For example, consider the source shown in Fig. 6.17(a). Note that $n^+(\{1\}) = glb\{a, b\}$ which does not exist. Also note that $n^-(\{2, 3\}) = lub\{c, d\}$ which does not exist. Therefore, we can define the upper and lower names of a set A as follows: $n^-(A) = maximal(\{t \in T \mid S(t) \subseteq A\})$ and $n^+(A) = minimal(\{t \in T \mid S(t) \supseteq A\})$. Consider for example the source shown in Fig. 6.17(b). Here we have:

$$\begin{aligned}
 n^-(\{1, 2, 3\}) &= maximal(\{c, d, e, b\}) = \{b\} \\
 n^+(\{1, 2, 3\}) &= minimal(\{b, a\}) = \{b\}
 \end{aligned}$$

The relationships obtained by the term-to-term mapping are less expressive than the relationships obtained by the term-to-queries mapping. For instance, suppose

term-to-term	$a \geq b$ $a \geq b'$	$a \leq b$ $a \leq b'$	
term-to-query	$a \sim b \vee b'$	$a \sim b \wedge b'$ $a' \leq b \vee b'$	$a \leq b \vee b'$ $a' \leq b \vee b'$

Fig. 6.18 Three examples

that we want to articulate the terms of the source S_1 in each one of the three examples that are shown in Fig. 6.18. The bottom part of that figure shows the articulation $a_{1,2}$ that is derived by the *term-to-term* articulation and the *term-to-queries* articulation in each of these three examples.

As an epilogue we could say that the ostensive method can be used for creating mappings not only between individual terms but also between arbitrary queries. The method is independent of the nature of the objects (i.e. the objects may be images, audio, videos). It can be implemented efficiently by a communication protocol and the common domain of the involved sources (or peers) is discovered during the execution of the protocol. Reference collections could be used for the case where the domain of sources are disjoint. For example in a peer-to-peer setting, each peer could index a small set of objects before joining the network. This will enable peers to run the articulation protocol on this reference collection.

Of course the method could be combined with other statistical methods. Other works aiming at finding a common language include [26].

6.4 Distributed Taxonomy-Based Sources

6.4.1 Mappings and Mediators

Suppose that we have different taxonomy-based sources and that we wish to provide a unified browsing or query interface to their indexed objects, either through one of the existing taxonomies or through a different one. The notion of *mediator* (initially proposed in [316]) can be adopted to this purpose. In brief, a mediator over a set of taxonomy-based sources consist of a taxonomy and a number of mappings between its taxonomy and the taxonomies of the underlying sources. These mappings aim at bridging the inevitable naming, granularity and contextual heterogeneities that may

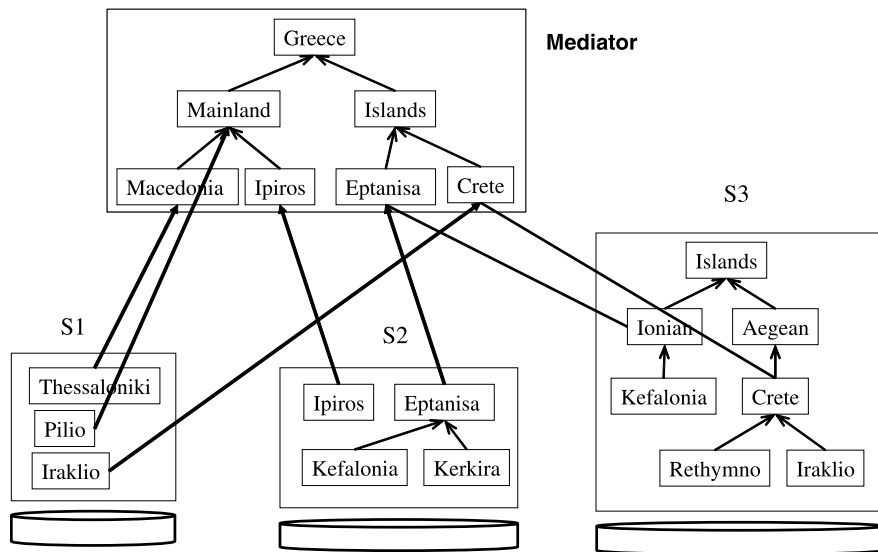


Fig. 6.19 A mediator over three taxonomy-based sources

exist between the taxonomies of the sources. These inter-taxonomy mappings can be defined either manually, automatically (e.g. using the *ostensive method* described in Sect. 6.3), or semiautomatically.

The mediator does not have to store object descriptions. The users of the mediator can pose queries using terms from the mediator taxonomy. By exploiting the inter-taxonomy mappings, these queries can be translated to queries that can be answered by the underlying sources. The answers returned by the underlying sources are then aggregated and are displayed to the user. Furthermore they can be summarized and explored according the interaction paradigm of dynamic taxonomies and faceted search. Users do not necessarily have to formulate queries, but they can directly start browsing using the taxonomy of the mediator. Figure 6.19 illustrates a mediator M over three taxonomy-based sources S_1 , S_2 and S_3 . The fat arrows that cross the taxonomy boundaries indicate relationships that are part of the mappings of the mediator. If we denote by \leq_M the subsumption relation of the mediator and by \leq_{MS_1} the mapping between the mediator M and the source S_1 , then in our example we have: $\text{Thessaloniki} \leq_{MS_1} \text{Macedonia} \leq_M \text{Mainland} \leq_M \text{Greece}$.

Integrating objects from several sources often requires *restoring the context* of these objects, i.e. adding information that is missing from the original representation of the objects which concerns the context of the objects. An example that demonstrates how the articulations can restore the context of the objects is shown in Fig. 6.20(b). The illustrated mediator provides access to electronic products according to the *type* of the products and according to the *location* of the stores that sell these products. The mediator has two underlying sources S_1 and S_2 , where the former corresponds to a store located in Heraklion, while the latter corresponds to a store located in Paris. The context of the objects of each source, here the location

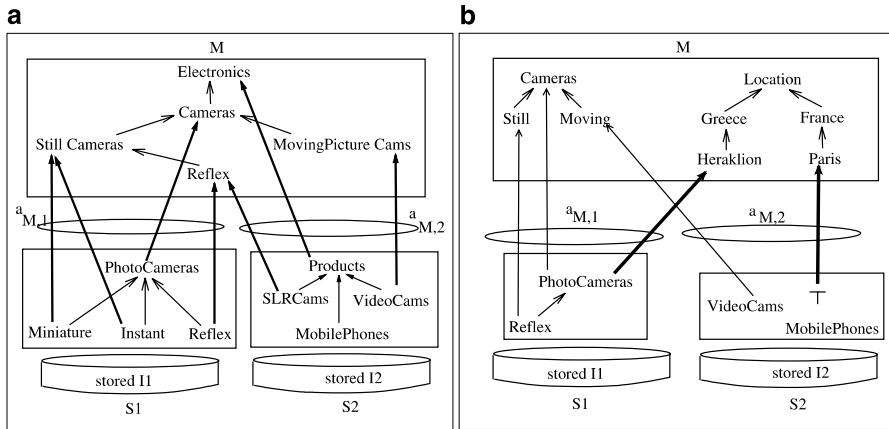


Fig. 6.20 Two examples of a mediator

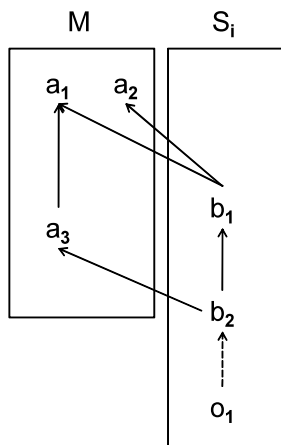
of the store that sells each product, can be restored by adding to the articulations appropriate relationships. Specifically, for defining that all PhotoCameras of the source S_1 are available through a store located in Heraklion, it suffices to put in the articulation $a_{M,1}$ the relationship $\text{PhotoCameras}_1 \leq \text{Heraklion}$, while for defining that all products of the source S_2 are available through a store located in Paris, it suffices to put in the articulation $a_{M,2}$ the following relationship $\top_2 \leq \text{Paris}$, where \top_2 denotes the maximal element of the subsumption relation of S_2 .

If the underlying sources return answers that are accompanied by their complete descriptions then the mediator can support the interaction paradigm of dynamic taxonomies and faceted search. Specifically, whenever an underlying taxonomy-based source (T_i, \leq_i, I_i, Q_i) (where $I_i : T_i \rightarrow \text{Obj}_i$) receives a query $q \in Q_i$, it returns the set of objects $\bar{I}_i(q)$ where each object is accompanied by the description $D_{\bar{I}_i}(o)$. The complete descriptions of the objects allow the mediator to compute the zoom points.

This can be explained using the example shown in Fig. 6.21. Suppose the mediator receives the query $q = a_1$. Subsequently the mediator sends the query $b_1 \vee b_2$ to source S_i . The source S_i then returns as answer the object o_1 (because $\bar{I}_{S_i}(b_1 \vee b_2) = \{o_1\}$). Note that $D_{I_i}(o_1) = \{b_2\}$, while $D_{\bar{I}_i}(o_1) = \{b_2, b_1\}$. The term b_2 allows the mediator to conclude that a_3 is a zoom-in point (because $b_2 \leq_{MS_i} a_3$) while the term b_1 in the complete description allows the mediator to conclude that a_2 is a zoom-side point (as $b_1 \leq_{MS_i} a_2$). Note that if S_i returned only the “direct” description (i.e. not the full description) of o_1 , then the mediator would not be able to infer that a_3 is a zoom-in point.

Consider a mediator M over k sources S_1, \dots, S_k , and k mappings $\leq_{MS_1}, \dots, \leq_{MS_k}$. If o is an object, let $D_i(o)$ denote the description of the object with respect to a source S_i (i.e. $D_i(o) = D_{\bar{I}_{S_i}}(o)$). The description of o with respect to

Fig. 6.21 An example of a mediator M over one source S_i



the mediator terminology T_M is given by:

$$D(o) = \bigcup_{i=1..k} DM_i(o), \quad \text{where } DM_i(o) = \{t \in T_M \mid t_i \in D_i(o) \text{ and } t_i \leq_{MS_i} t\}$$

The interested reader can refer to [292] for more details.

So far we have seen a virtual integration approach. In a *materialized integration approach* the integrated system by querying the underlying sources it gets and stores the extension of all of its terms. In that case, it would be enough for the sources to return objects with their (direct descriptions).

Query evaluation algorithms for several mediator operation modes are given in [306]. By combining different modes of query evaluation at the sources and the mediator, and different types of query translation, a flexible, efficient scheme of mediator operation is obtained, which can accommodate various application needs and levels of answer quality.

6.4.2 Distributed Query Evaluation

The same integration problem in the context of a *peer-to-peer* setting consisting of primary sources, mediators, and articulated sources (as illustrated in Fig. 6.22), is elaborated in [185]. The difference between the P2P architecture and the classical two-tiered mediator approach is that in a P2P system the mappings between the peers may lead to cyclic dependencies between the query evaluation tasks of the peers. Such cases require special treatment in order to avoid endless query evaluation and to optimize the evaluation of queries.

One way of interpreting such a network is to view it as a single source which happens to be distributed along several sources, each dealing with a specific subterminology of the network terminology. The global source can be logically reconstructed

Fig. 6.22 Peer-to-peer systems of taxonomy-based sources

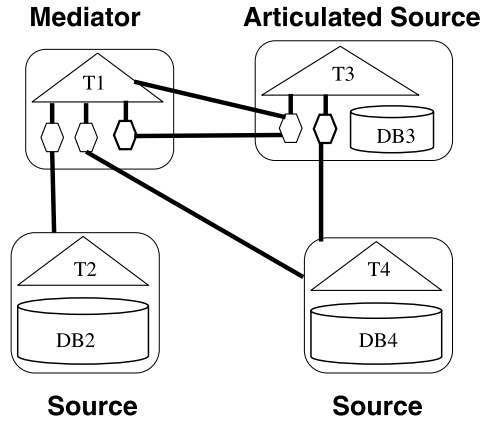
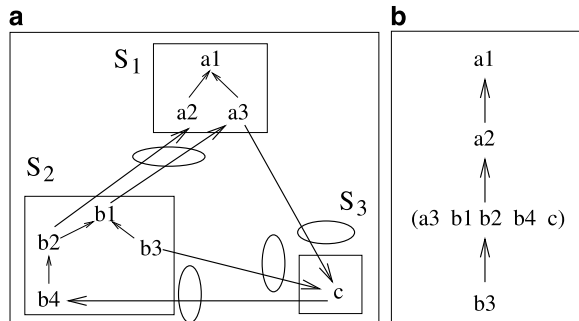


Fig. 6.23 A network of three sources and the inferred global taxonomy



by removing the barriers which separate local sources, as if (virtually) collecting all the network information in a single repository. Note that this global source does not preexist. It emerges in a bottom-up manner by the mappings of the peers. This is one difference that distinguishes peer-to-peer systems from federated distributed databases. For example, Fig. 6.23(a) shows a network of three sources. Notice that although the subsumption of each source is acyclic, the global subsumption has a cycle:

$$a3 \leq_{S1} c \leq_{S3} b4 \leq_{S2} b2 \leq_{S2} b1 \leq_{S3} a3$$

Figure 6.23(b) shows the global subsumption relation. The notation (a3 b1 b2 b4 c) means that these terms are equivalent with respect to the global subsumption relation. Query answering and browsing should be based on this relation.

An arsenal of query evaluation algorithms for pure peer-to-peer systems for computing sound and complete answers are presented in various works including [185, 297].

Specifically, [297] identified two basic query evaluation approaches: one based on *query rewriting*, the other based on *direct* query evaluation. For each approach a *centralized* and a *decentralized* algorithm for carrying out the query evaluation task was given.

[185] approached the same problem by mapping sources into Datalog programs. Several cases are examined along the following axis:

- taxonomy type
Taxonomies may comprise single terms, or complex concepts, e.g. relationships of the form $(Animal \wedge FlyingObject) \vee Penguin \leq Birds$.
- query type (with negation or not), and
- mappings type (term-to-term, term-to-query, query-to-query).

For reasons of space, here we just mention a few of the results. In brief, if negation appears in term-to-query mappings then query answering is coNP-hard. The same holds even if we have not negation but we have query-to-query mappings. More recent and detailed results are reported in [186].

6.5 Synopsis and Bibliographic References

Regarding the description of the valid compound terms in an intensional manner, [299] allowed only positive or only negative statements, while CTCA (originally presented in [301] and in more complete manner in [308]) combined these views. The comparison of CTCA with Description Logics is analyzed in [305], while [21] shows how CTCA can be efficiently represented using logic programming, providing an alternative kind of implementation.

A formulation of the index improvement problem (as described in Sect. 6.2) founded on abduction is described in [184, 187].

Regarding inter-taxonomy mappings, there are several works on the general problem of constructing mappings e.g. see [223]. An extension of the notion of naming functions (as described in Sect. 6.3) for ordered sets is described in [298].

Acknowledgements Most of the work presented in this chapter is joint work with Anastasia Analyti, Nicolas Spyrtos, Carlo Meghini, and Panos Constantopoulos.

Chapter 7

Taxonomy Design

Wisam Dakka, Panagiotis Ipeirotis,
and Giovanni Maria Sacco

*“Think simple” as my old master used to say—
meaning reduce the whole of its parts into the
simplest terms, getting back to first principles.*

Frank Lloyd Wright, 1867–1959

This chapter discusses the design of taxonomies to be used in dynamic taxonomy systems. Although the only actual requirement of dynamic taxonomies is a multidimensional classification, i.e., objects are classified under more than one concept, an organization by facets is normally used.

In the first section, we provide guidelines for the design of DT taxonomies, which include the automatic construction from structured data, and the retrofitting of traditional monodimensional taxonomies. In the second section, we show how a faceted taxonomy can be automatically extracted from the infobase itself when objects are textual or are described by textual captions or tags.

7.1 General Guidelines for Taxonomy Design

The extensional inference rule of dynamic taxonomies has important implications on conceptual modeling. Inference simplifies taxonomy creation and maintenance because concept relationships are dynamically inferred, whereas all the possible relationships need to be described in traditional taxonomies. In addition, the inference rule coupled with conceptual summaries makes the relationships between different concepts immediately visible to the user. Finally, dynamic taxonomies can synthesize compound concepts so that they do not need to be explicitly represented in the taxonomy. This removes the principal cause of the combinatorial growth of traditional taxonomies.

These properties of dynamic taxonomies suggest to break down the conceptual taxonomy to a minimal set of basic constituent concepts or *fundamental facets*, whose combinations can fully describe the entire universe of discourse. This operation closely resembles normalization in relational database systems, because its goal is the reduction of (conceptual) redundancy, and the factoring of common properties. The reduction of redundancy minimizes the complexity of the resulting taxonomy.

At the same time, factoring common properties improves the exploration expressivity of the navigation, because it substitutes complex concepts with relationships among primitive concepts.

The principal guideline [236] is to organize the taxonomy as a set of independent, ‘orthogonal’ subtaxonomies (facets or perspectives), which behave as coordinates in a multidimensional space. Taxonomies produced in this way are ‘minimal’ without a decrease in expressivity. Such an organization, it must be stressed, is a design guideline, and not a requirement of the model, because dynamic taxonomies only require a multidimensional classification.

We can make the notion of ‘orthogonal’ facets clearer by recalling the analysis in Sect. 3.2. In a faceted organization, we have the fastest convergence when each facet F is independent of any other facet F' , $F' \neq F$, i.e., when given any two concepts $A < F$ and $B < F'$, $p(A|B) = p(B|A) = p(A)p(B)$. Such an independence can only be verified on the extension, and this criterion is therefore *a posteriori*. However, it can be used in design by checking if the underlying independence hypothesis is likely to hold in practice.¹

The practical criterion in arriving at a set of facets is to identify a set of single criteria which can be used to subdivide a given concept (initially, the universe of discourse) [276].² For example, a digital camera can be characterized by a set of single criteria such as ‘Price’, ‘Weight’, and ‘Resolution’.

In addition to facets, we can usually define primitive concepts which partition the corpus into disjoint sets. For example, a broad-scope corpus like an encyclopaedia can have primitive concepts like ‘art’, ‘science’, and ‘history’. These concepts, which are *partitioning aspects*, are different from facets, such as ‘location’ or ‘time’. Facets, which are rather *cross aspects* and tend to cover the entire corpus, rather than partitioning it.

Once the fundamental concepts of the universe of discourse have been identified, each will be described by a subtaxonomy, i.e., a hierarchy of subsumptions, typically specializations. Faceting and IS-A hierarchies are two independent conceptual devices, though both can be represented by subsumptions. In an IS-A hierarchy, the extension of a child of a concept C is generally a subset of the extension of C ; the union of the extensions of the children of C is usually equal to the extension of C . In a subdivision of C by facets, the extension of each facet (e.g., Price) is usually equal to the extension of C , as the digital camera example above shows.

The ‘faceting’ process can be repeated again at each level, but doing so constructs a static decision tree and consequently decreases the exploratory flexibility of DT

¹Statistical facet independence substitutes Ranganathan’s Canon of Concomitance [226] and the Classification Research Group (CRG) Principle of Mutual Exclusion [125, 276] in traditional facet analysis, which state that no two facets can overlap in content. Content overlap is determined on the basis of concept labels, and is therefore imprecise and not easily applicable to DTs where concepts are abstract.

²This is CRG’s Principle of Division [125].

access.³ In our context, faceting is generally used once for determining the root nodes of each subtaxonomy. There are however cases in which subsequent faceting can be used to simplify the taxonomy, and consequently user access. Assume we want to create an e-catalog for heterogeneous product types. Each of these product types⁴ (i.e., cameras, refrigerators, tv sets) will have a set of disjoint features, with Price being possibly the only feature common to them all.⁵ In this case, we can have a first subdivision by product type, and a second one by specific features (e.g., resolution, energy class).

Traditional facet analysis uses multiple principles to guide the faceting process. The most important ones, the Principle of Division and the Principle of Mutual Exclusion, have already been discussed. The Principle of Relevance, i.e., that division is performed according to criteria that are useful for access, clearly applies.

It is important to stress that the other two principles used by traditional facet analysis, the Principle of Ascertainability and the Principle of Permanence, do not apply in our present context. The Principle of Ascertainability requires that the division criterion must be *always* ascertainable: Ranganathan suggests that ‘date of death’ should not be used as a facet, because it is impossible to know when people will die [276]. This is obviously related to the problem of null values, and we can apply common solutions such as using special facet values like ‘unknown’, ‘not applicable’, etc., or simply not classifying under a facet if no child value applies.

The Principle of Permanence, i.e., facets should represent characteristics of division which represent permanent qualities of the concept being divided, is the legacy of the static physical medium underlying facet theory. In a traditional library, the classification of a book cannot change. In our present context, we have no problem in changing it, and this can be extremely useful in certain application areas such as e-auctions. In e-auctions, a facet ‘Time to completion’, which is not permanent, can be a fundamental access path.⁶

The extensional inference rule allows to represent *multiple inheritance* either explicitly (by a specific concept) or implicitly (by extensional inference). Consider the classic ‘working student’ example. A working student has a multiple inheritance because he inherits his properties from both ‘worker’ and ‘student’. We can represent this explicitly in the intension, by having a specific ‘working student’ concept in the taxonomy, and make it a child of both ‘student’ and ‘worker’. The visual interface remains the same, because the user will see ‘working student’ if he opens ‘student’, and ‘working student’ if he opens ‘worker’. Although ‘working student’ appears in different places in the taxonomy, it is actually the same concept, represented by the same internal id, and consequently identifies the same extension.

³This is actually prescribed by facet analysis where the fixed, predefined order used for division is called the ‘citation order’ and actually defines a static decision tree. This is one of the most evident differences between traditional faceted approaches and DTs.

⁴These are partitioning aspects.

⁵Price is a cross aspect.

⁶See also Sect. 8.1.7.2.

As an alternative, this multiple inheritance can be represented implicitly, by simply classifying every working student under ‘worker’ and ‘student’. This assumes that the existence of a multiple inheritance is captured by the classification system. In the case of multiple inheritance (and of any compound concept), the avoidance of compound concepts is an opportunity rather than a rule. Although the need for a compound concept often indicates that the taxonomy was not correctly defined, we advise to use an explicit representation (specific concepts) when the topic is specific, well-known, or deserving user attention [236]. For example, Internet is best described explicitly by a specific concept, rather than implicitly by the intersection of ‘computers’ and ‘communication networks’.

With respect to the ‘shape’ of the taxonomy, human factors dictate some restrictions on the number of levels and on the average number of children for each node. We believe that a number of children larger than approximately 10 objects makes the selection of the appropriate child(ren) to zoom on too difficult [236]. At the same time, an average number of levels larger than 3 or 4 is likely to result in taxonomies that require too much effort in order to access the terminal level and are difficult to understand. These two constraints place the number of terminal concepts for effective DTs roughly between 1,000 and 10,000.

Concepts with a large cardinality can cause an inordinate growth of the taxonomy. Such concepts include concepts with numeric values (prices, weights, etc.), with dates (start date, birth date, etc.), and concepts such as Persons, Companies, etc. Numeric and date concepts can be represented by ranges rather than by specific values, but this causes a loss of information.

If the corresponding data is available in a database, the designer can opt for an asymmetric approach and use an external query method (e.g., an SQL query) to focus on specific values, and summarize the result through a reduced taxonomy. Obviously, this approach does not allow to summarize focus sets through these concepts, and should be consequently avoided if such summaries are important. For example, in an infobase describing classical music recordings, it would probably be a bad idea not to represent composers in the dynamic taxonomy, but only in an external database. A focus on ‘String Quartets’ cannot be summarized by composer, which is an important navigational dimension in this application.

An alternate approach, which is often more appropriate and is discussed in Sect. 5.3, is the use of *virtual concepts* to virtualize part of the taxonomy. Virtual concepts appear to the user as *bona fide* concepts, but they are synthesized on demand from external concepts.

Although DTs taxonomies are usually balanced trees, it sometimes beneficial to allow unbalanced subtrees in order to have terminal concepts with roughly the same selectivity, and consequently less variance in the cardinality of concept intersections [236]. Since the selectivity of a terminal concept depends on the extension, rather than on the intension, of the infobase, it is useful to periodically monitor the extension. A high number of objects classified under a terminal concept *C* usually indicates that a further refinement of *C* (increasing the abstraction level) is advisable. Conversely, a very small number of object classified under *C* indicates that *C*’s specializations may be discarded and documents directly classified under *C*, thereby decreasing the abstraction level.

However, increasing the abstraction level is

1. not always possible, because concepts are not infinitely specializable; and
2. not strictly required, because concept intersections considerably reduce the problems deriving from skewed distributions.⁷

A decrease in the abstraction level is useful to simplify the taxonomy and to reduce its storage requirements. The actual impact on user interactions is generally negligible because, especially if related counters are shown, the user will not expand the concept.

These considerations and our practical experience, suggest that dynamic taxonomies tend to be stable, and that schema update and maintenance is a relatively rare event, in practice, after the initial design and test phase. Base, fundamental concepts evolve slowly in time. The real dynamic part of a DT is given by the relationships among concepts, which often change very rapidly. However, these relationships are dynamically computed through the extensional rule, and do not require any change in the taxonomy.

Concept labels should be clearly understandable and unambiguous, and taxonomic abstractions should be clearly perceivable and consistent. Since the taxonomies designed according our guidelines minimize the number of required concepts, this task is easier than in traditional approaches. In addition, in DTs the meaning of any concept can be made clear by examples not only in the form of sample objects, but also in terms of summaries which highlight related concepts [236].

Finally, children of concepts must be arranged in a clear, consistent way. In this context, we can refer to traditional facet analysis [276] which proposes the Principle of Relevant Succession⁸ for child order. This principle identifies the following ordering strategies:

1. *Chronological Order*;
2. *Alphabetical Order*;
3. *Spatial/Geometric Order*, which orders children by contiguity according to seven possible ways: Bottom Upwards, Top Downwards, Left to Right, Clockwise Direction, Counter-Clockwise Direction, Periphery to Center, and Center to Periphery;
4. *Simple to Complex Order vs. Complex to Simple Order*;
5. *Canonical Order*, which is an established, traditional order for the subdivision of a concept. For example, the canonical order for the children of Philosophy is Logic, Epistemology, Metaphysics, Ethics, Aesthetics, etc.;
6. *Increasing Quantity vs. Decreasing Quantity*

Ranganathan also includes ordering by ‘literary warrant’, which lists children by decreasing number of objects listed under each concept. A similar, but more general, notion was used in Sect. 5.7.3. This is a dynamic ordering scheme, and its variation over time may disorient users.

⁷See Sect. 3.2.6.

⁸Also known as Ranganathan’s principles for helpful sequence.

The guidelines discussed above are generally valid for all DT applications. Taxonomy design for structured data is clearly a design ‘in the small’, i.e., a design for a limited, well-defined application domain. For example, the catalog of an e-shop is usually already available as a relation in a RDBMS. The focus here is on the automatic creation of taxonomies from structured data.

Taxonomy design for unstructured data (e.g., free text, images) is, at least potentially, a design ‘in the large’, in that it can be applied to describe the entire present (and future) knowledge. Often, these applications already use a monodimensional taxonomy, which can be transformed to a faceted multidimensional taxonomy as we discuss in the following.

7.1.1 Design ‘in the Small’

7.1.1.1 Automatic Construction for Views

To illustrate the basic design principles, we will consider the currently most frequent application of dynamic taxonomies: the intelligent exploration of a relation, or, more generally, of a relational view V , which can be derived from base relations and can also obviously represent the temporary result of a query. This structure is inherently multidimensional: each attribute in the view is an independent indexing dimension. Each tuple in the view is a DT object, which is classified in the taxonomy according to the values of its attributes.

The creation of a bare-bones shallow dynamic taxonomy from a relational view V can be accomplished as follows:

```
for each attribute  $a$  in  $V$ ,
    create a facet  $f(a)$  in the taxonomy
for each value  $v$  in the domain of  $a$ 
    create a son  $v$  of  $f(a)$  in the taxonomy
```

Each record r in V is then indexed by taking, for each attribute a in V , its value $r.a$ and indexing r under the corresponding son of $f(a)$. Since a concept that has no object classified under it is automatically pruned from the initial taxonomy, we only need to create for each facet $f(a)$ as many sons as the unique values in the attribute a in V .

This algorithm establishes a fundamental correspondence between facets and attributes of a relational view, which helps to put the guidelines described above in a more essential perspective. Interestingly, design techniques for relational databases, which are based on the Entity–Relationship model [64], consider the selection of attributes (facets) as an ‘intuitive’ initial step, and the design guidelines for it are likely to be “each property of interest of a specific entity is represented by an attribute”. This indicates that the fundamental principle for facet definition is really a principle of relevant division: each attribute inherently defines a division criterion, and only relevant attributes are defined.

In considering the mapping of a general relational view to a dynamic taxonomy, there are multiple considerations. First, the integration between dynamic taxonomies access and access through any other retrieval method (database or XML queries, in the present context) can be exploited, to reduce the number of facets. Not all the attributes in the view must necessarily have a counterpart in the dynamic taxonomy. However, attributes to be summarized must be explicitly described in the taxonomy.

Second, in the approach proposed above, attribute values are enumerated in the taxonomy as immediate children of a facet. While this is acceptable for ‘small’ enumerative domains, such as Brand, Country, etc. it becomes rapidly unwieldy as the number of different values increases. In addition, such a flat representation does not allow any systematic exploration. As an example, a global company would probably find a hierarchical grouping of locations into continents, nations, counties, etc., more useful than a flat list of locations. The same rationale holds for domains, such as numeric domains, whose potential number of different values is infinite, and are usually more manageable from the user point of view by structuring them in (multi-level) ranges of values. Consequently, in general, an independent subtaxonomy is defined for each attribute in order to structure existing values in a meaningful way.

Chakrabarti et al. [61] categorize results of relational queries by generating a summary multi-level taxonomy on the fly. The taxonomy generated is a plain taxonomy, so that their approach is a special case and considerably less powerful than dynamic taxonomies. The interesting point in their work is that the taxonomy is generated in such a way as to minimize a formally defined information overload cost for users. The adaptation of the higher level of the taxonomic tree is especially valuable as a way to define meaningful ranges of numeric values. As an example, consider the catalog of a megasite carrying everything from pins to elephants.⁹ Prices in such a store would exhibit such a large variance that it would be difficult to predefine value ranges which are meaningful for all users.

However, taxonomies generated in this way are not generally applicable because they do not necessarily capture the semantics of abstraction: e.g. the fact that Rome is in Italy and that Italy is in the European Union, which we contend is fundamental in most applications. An additional weakness of this method is that different query results produce different taxonomies, thereby disorienting the user.

These considerations suggest that a correct design for a general taxonomy should comprise both explicitly and implicitly defined concepts. Facets whose concepts can be taxonomically arranged in a hierarchical way, to model IS-A or PART-OF relationships, must be explicitly defined in the taxonomy. An example is a Location facet which can be organized by nations, continents, etc. We note here that more than one facet can correspond to a given attribute. For instance, Locations can be semantically structured in different ways: by nations and continents, or by organizations (NATO, OPEC), or by climate, etc. On the other hand, attributes with a very high number of different values and shallow facets (i.e., facets whose sons are ac-

⁹London’s Harrods motto.

tual attribute values) are best represented in an implicit way by queries on the view itself, as described in Sect. 8.1.

7.1.1.2 Dynamic Taxonomies for E–R Schemata

The mapping strategy outlined above is simple and effective, and has some affinities with star schemata in OLAP applications [62]. However, it consolidates all attributes together in a possibly large number of facets. This is undesirable because it makes user orientation difficult: a taxonomy with tens of facets would pose a cognitive challenge to most users.

A natural way of structuring the taxonomy in such a way as to make it easily understandable by users, is by using an Entity–Relationship schema¹⁰ as a starting point. An E–R schema structures the infobase as entities which represent real-world object types, and their relationships. Entities in the schema therefore provide natural top-level facets for the taxonomy, with the respective attributes appearing as immediate sons. The objects to be classified in the dynamic taxonomy are the tuples in the universal relation view constructed for the schema.

Regarding relationships, a first viable strategy is to represent them in the same way as entities, i.e., each relationship defines a top-level facet, with only the attributes of the relationship appearing as immediate sons. A relationship with no additional attributes is not explicitly represented in the schema. Participating entities need not be explicitly represented within the context of the relationship because the extensional inference rule establishes the right relationships between the top-level facets which represent the entities involved.

Consider the schema in Fig. 7.1 in which the primary keys of entities are assumed to be represented by surrogates (i.e. unique identifiers). This schema can be mapped into the following relational schema:

```
Part(Part#, Pname, Ptype)
Supplier(Supplier#, Sname, Slocation)
Plant(Plant#, PLname, PLlocation)
PlantUsesPart(Plant#, Part#)
SupplierSuppliesPart(Supplier#, Part#, Price)
```

First, we construct the universal relation view for this schema:

```
UR(Part#, Plant#, Supplier#, Pname, Ptype, PLname,
  PLlocation, Sname, Slocation, Price)
```

by outer-joining all the entities and relationships in the schema.

¹⁰The Enhanced Entity–Attribute model [97], or other extensions of the E–R model to conceptual hierarchies, are more adequate to the task at hand, but not as well known, and more complex.

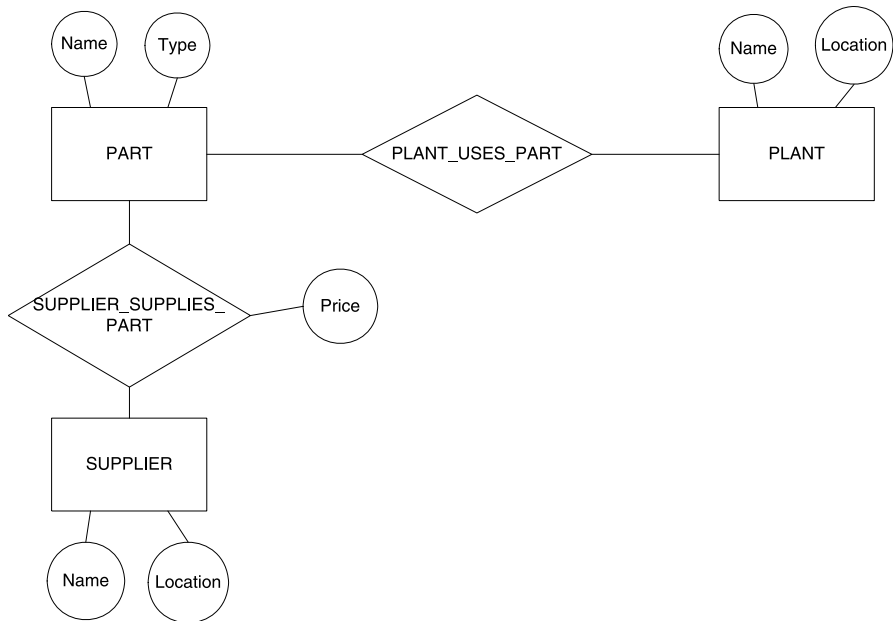


Fig. 7.1 Sample E-R schema; entities are implicitly identified by surrogates

This means that each object in the DT will be identified by a triplet (Part#, Plant#, Supplier#) such that Part# is supplied by Supplier# and/or Part# is used by Plant#. Both Plant# and Supplier# can be NULL.

Second, we decide which attributes will be described in the dynamic taxonomy, whether subtaxonomies are needed for selected attributes, and whether concrete or virtual concepts are to be used.

Third, we make entities and relationships explicit in the schema. A candidate dynamic taxonomy schema for the example is reported in Table 7.1.

As an example of use, assume that we zoom on a specific supplier S . The objects we are selecting are all those tuples in UR, such that $\text{Supplier\#} = S$. The reduced taxonomy will report all the Parts supplied by S , and in addition all the Plants which are using such Parts (and their appropriate attribute values). In addition, also the Prices for Parts supplied by S will be reported.

The taxonomy reported in Table 7.2 presents an alternate strategy for representing relationships. Here, all the relationships are explicit top-level facets in the taxonomy and, in addition, all participating entities are explicitly represented in the context of each relationship as immediate sons.

This taxonomic schema shows a very important point: although the extensional inference rule infers unnamed relationships among concepts, the meaning of specific relationships can be made concrete and visible to the end-user. In this schema, relationship facets are used to disambiguate the unnamed relationships inferred by the extensional inference rule. When zooming on $\text{Part} > \text{Name} > \text{XYZ}$, the extensional inference rule establishes relationships among entity instances; the two facets repre-

Table 7.1 Taxonomy for sample E–R schema

Part
Name
Type
Mechanical
Electric
Electronic
Supplier
Name
Location
Africa
America
Asia
Plant
Name
Location
Africa
America
Asia
SupplierSuppliesPart
Price

senting relationships disambiguate the role of the part: whether it is a used-by-plant part or a supplied-by-supplier part or both. At the same time, if the user zooms on PlantUsesPart>Part>Name>XYZ, he specifies a specific role for XYZ. No role is specified by Part>Name>XYZ.

In summary, a top-level facet representing an entity represents such entity in any role, whereas a specific role is specified when this same facet is set as the son of another facet.

By converse, there might be attributes which are shared among different entities: in the current example, Location is an attribute both of Supplier and Plant. Instead of representing these attributes only as sons of the facets representing their entities, it is more convenient to add a top-level facet for each of them. This allows the user to zoom on a specific value of that attribute regardless of its role, i.e., of the entities to which the attribute is associated. In the examples in Tables 7.1 and 7.2, this means adding a top-level facet, Location: the user zooming on Location>Africa will select Plants and Suppliers in Africa.

Similar considerations apply to domains, and in particular to dates. A facet representing dates would allow the user to zoom on a specific date, and have a summary of all the entities and relationships related to that date. Obviously, this technique should be applied to attributes and domains only if focusing on them in a role-free way is useful for the user. Otherwise, the additional facets needed only make the taxonomy more complex and harder to understand.

Table 7.2 Alternate taxonomy for sample E-R schema

Part
Name
Type
Mechanical
Electric
Electronic
Supplier
Name
Location
Africa
America
Asia
Plant
Name
Location
Africa
America
Asia
PlantUsesPart
Plant
Name
Location
Africa
America
Asia
Part
Name
Type
Mechanical
Electric
Electronic
SupplierSuppliesPart
Supplier
Name
Location
Africa
America
Asia
Part
Name
Type
Mechanical
Electric
Electronic
Price

Dynamic taxonomies represent an intermediate model between traditional taxonomies and complex semantic models. Dynamic taxonomies are more powerful than plain taxonomies because traditional taxonomies only describe subsumptions, whereas dynamic taxonomies are able to represent, in a dynamic way, any kind of relationship that can be inferred from empirical evidence, that is from the classification itself.

Dynamic taxonomies are less powerful than general semantic networks or semantic data models, because these additional relationships are, in general, unnamed and therefore ambiguous. However, we have shown that the meaning of unnamed, inferred relationships can be made explicit by a careful design of the dynamic taxonomy. From the user point of view, both traditional and dynamic taxonomies are easily understood by end-users, whereas general semantic schemata are not. Whenever user access is important, the use of dynamic taxonomies which represent complex semantic schemata appears beneficial.

Dynamic taxonomies which are flexible and easily understandable by end-users can be derived from relational views and E–R conceptual schemata. If a database or a semantic information base already exists, the design methodology produces a dynamic taxonomy which captures the semantics of the information base and makes it easily available to end-user. Even if no schema exists, starting with traditional and well-understood data design techniques and applying the methodology we introduced will produce consistent and effective dynamic taxonomies that are at the same time exhaustive and easy to understand, even for demanding applications.

7.1.2 Design ‘in the Large’

The design of taxonomies ‘in the large’ is usually required for unstructured corpora (e.g., free text, images) with a very broad application domain. Examples include encyclopaedias, news feeds, very large image bases, catalogs of WWW resources, etc. In this context, design can be carried out through the general guidelines described above. Since the substantial equivalence between facets and E–R entities and attributes indicates that fundamental facets depend on the application domain, there are no predefined sets of facets for corpora with broad domains. Or, more precisely, no fixed set of facets is likely to be acceptable for all broad domains.¹¹

This does not necessarily mean that it is impossible to give indications for the selection of fundamental facets and partitioning aspects for very broad domains. Certainly, ‘Space’ (intended as location) and ‘Time’ (intended as Chronology) belong to the set of fundamental facets, as they are immediately relevant to most real-world objects. Ranganathan proposed five fundamental categories [226] to describe the entire universe of ideas:

- P (Personality, or Who): what the object is primarily ‘about’. This is the ‘main facet’;

¹¹This view is accepted also by many researchers in Information Sciences [276].

- M (Matter, or What): the material of the object;
- E (Energy, or How): the processes or activities which take place in relation to the object;
- S (Space, or Where): where the object happens or exists;
- T (Time, or When): when the object happens or exists.

The Bliss classification system [191] considerably extends the list of fundamental facets:

- thing
- kind
- part
- property
- material
- process
- operation
- patient
- product
- by-product
- agent
- space
- time

and defines the following partitioning aspects:

- Generalia, Phenomena, Knowledge, Information science & technology
- Philosophy & Logic
- Mathematics, Probability, Statistics
- General science, Physics
- Chemistry
- Astronomy and earth sciences
- Biological sciences
- Applied biological sciences: agriculture and ecology
- Physical Anthropology, Human biology, Health sciences
- Psychology & Psychiatry
- Education
- Society
- History
- Religion, Occult, Morals and ethics
- Social welfare & Criminology
- Politics & Public administration
- Law
- Economics & Management of economic enterprises
- Technology and useful arts
- The Arts
- Music
- Language and literature

Table 7.3 Dewey
classification fragment

800	Literature and Rethoric
	...
810	American literature in English
811	Poetry
812	Drama
813	Fiction
814	Essays
815	Speeches
816	Letters
817	Satire and humor
818	Miscellaneous writings
820	English literature
821	Poetry
822	Drama
823	Fiction
824	Essays
825	Speeches
826	Letters
827	Satire and humor
828	Miscellaneous writings
830	German literature
831	Poetry
832	Drama
833	Fiction
834	Essays
835	Speeches
836	Letters
837	Satire and humor
838	Miscellaneous writings
840	French literature
841	Poetry
842	Drama
843	Fiction
844	Essays
845	Speeches
846	Letters
847	Satire and humor
848	Miscellaneous writings

Other fundamental facets and partitioning aspects can be suggested by different sources, such as WordNet [100], Wikipedia at wikipedia.org, and the Open Directory Project at dmoz.org.

In many practical cases, the corpus might be already described by a taxonomy, but such a taxonomy is likely to be a traditional, monodimensional taxonomy. Monodimensional classification schemes are intuitively a bad idea. It is very difficult to find examples where only a single dimension or feature can be used to classify items. In fact, monodimensional schemes such as the Dewey classification for libraries [87] ‘linearize’ a multidimensional scheme into a monodimensional one. To do so, compound concepts are created and used. As an example, refer to the Dewey classification fragment in Table 7.3.

The reader will note that the entries are the cross product of two sets of label terms: {English, German, French, ... } and {poetry, drama, fiction, ... }. The first set represents the ‘Language’ facet, whereas the second set represents the ‘Literary Genre’ facet. Now the same fragment can be reorganized by facets as in Table 7.4. The retrofit of a monodimensional taxonomy to a faceted taxonomy is usually a fairly straightforward process, which mainly involves finding common terms in concept labels and factoring them out. The factoring process is usually simpler and more accurate if fundamental facets are preliminarily isolated. This allows one to disambiguate polysemic terms such as English, which means ‘written in English’ in ‘English poetry’, and ‘located in England’ in ‘English history’.

The advantages of the resulting faceted taxonomy are

1. the minimization of concepts, which decrease from 32 concepts in the Dewey fragment to 12 in the faceted fragment, i.e., from $n \cdot m$ to $n + m$;
2. an easy, symmetric correlation between features. If the user focuses on ‘drama’, she will find that there are dramas in different languages; if she focuses on ‘English’, she will find the different literary genres for English, including ‘drama’.

Table 7.4 Faceted classification fragment

Language
American English
English
German
French
Literary Genre
Poetry
Drama
Fiction
Essays
Speeches
Letters
Satire and humor
Miscellaneous writings

the monodimensional taxonomy, access is asymmetric, and in the example only the second type of access is allowed;

3. the combinatorial complexity of compound concepts in a monodimensional taxonomy forces the taxonomy designer to a biased view of the universe. In the Dewey classification, each ‘major’ language has eight literary genre descriptors (poetry, drama, fiction, essays, speeches, letters, satire and humor, and miscellaneous writings). ‘Minor’ languages, such as Portuguese or Romanian have only one descriptor, and all the languages in East and Southeast Asia are grouped together into a single descriptor. It is easy to imagine that this will not be the perspective of a Portuguese or Thai classifier or user.

7.2 Automatic Construction from Text Information Bases

Faceted searching and browsing can be improved by utilizing various facets. However, in the presence of many facets, we have to choose which ones to present to the user. Presenting tens or hundreds of facets will make information access more difficult rather than easier. Hence, we have to select only the few that will be most useful for browsing purposes. For example, we would identify and assign video clips from a YouTube collection to the “Animals” or “Location” facets. Then, for each item in the collection, we would supply text-annotated keywords to describe the relationship between the item and the facet to which it has been assigned. Finally, we would use these text-annotated descriptions to construct faceted hierarchies for browsing the collection or lengthy search results. This chapter is dedicated to automation of this task to support wide deployment of faceted hierarchies over textual and text-annotated collections. One example of a text-annotated collection is the Corbis royalty-free image collection. Corbis has a large set of annotated images, in which each image has a title, free-text description, and a set of keywords associated with it. Each keyword is manually assigned by the Corbis annotators to one of the 38 facets that Corbis uses. The New York Times archive is an example of a large textual collection of news articles, dating to 1851.

In this chapter, we present methods to automatically discover facets and their useful browsing terms from a collection. In Sect. 7.2.1, we give a detailed overview of the problem of finding useful facet terms, and in Sects. 7.2.2 and 7.2.3 we describe supervised and unsupervised methods for extracting such facets from collections with textual and text-annotated data. In Sects. 7.2.4 and 7.2.5, we evaluate our approaches over two different textual and text-annotated collections. Finally, we elaborate on future work in Sect. 7.2.6 and conclude our chapter in Sect. 7.2.7.

7.2.1 Problem Overview

One of the bottlenecks in the deployment of faceted interfaces over collections of text or text-annotated documents is the need to *manually* identify useful dimen-



Fig. 7.2 A Flickr image shot by Sephiroty Fiesta

sions or facets for browsing a collection or lengthy search results. Once the facets are identified and assigned to the collection objects through descriptive keywords, a hierarchy is built and populated with the collection objects to enable the user to locate the objects of interest through the hierarchy. Static, predefined facets and their manually or semi-manually constructed hierarchies are usually used in commercial systems such as Amazon and eBay, with their faceted hierarchies for consumer products. The first step to automate the construction of faceted hierarchies is to identify the facets that are useful for browsing and assign them to the collection objects. In this chapter, we focus on this step.

We consider two types of collections in this chapter. A collection of the first type consists of objects that have some associated descriptive keywords or tags. One example of this type of collection is the Corbis royalty-free image collection. Corbis has a large set of annotated images, in which each image has a title, a free-text description, and a set of keywords associated with it. Each keyword is manually assigned by the Corbis annotators to one of the 38 facets that Corbis uses. Other examples include Flickr and YouTube. In contrast, a collection of the second type consists of free-text objects such as news articles in The New York Times archive or Newsblaster.

These two types of collections need to be processed differently for facet extraction, as the following example illustrates:

Example 7.1 Consider the Flickr image in Fig. 7.2 of a dog and a cat on a rocky beach. Typically, in Flickr-like collections each image (object) is tagged with several descriptive keywords. The image in Fig. 7.2 is associated with keywords “dog”, “cat”, “sea”, “beach”, “rock”, and “Tampico”. As we can see, these keywords describe several orthogonal aspects (facets) of the image: some describe the “Ani-

mals” in the image, some describe the “Topographic Features”, and others describe the “Location” where this image was taken. Knowing which keyword belongs to which facet is key for generating a meaningful browsing hierarchy for each facet. Unfortunately, we often do not have these keywords organized across facets and we resort to generating a single hierarchy that fits all keywords in a large collection such as Flickr or YouTube. A single hierarchy of this type is typically awkward and confusing for browsing. Collections such as The New York Times archive bring an additional challenge: news stories usually are neither associated with descriptive keywords nor organized across facets. Therefore, we have to resort to the story text to identify descriptive keywords and organize them across facets.

In this chapter, we investigate an extraction technique for each collection type. First, for the text-annotated collections (e.g., Flickr) that already have some keywords organized across different facets, we can use this facet data to train a machine learning algorithm to classify keywords in the appropriate facets (Sect. 7.2.2). The classifier can then be used to assign keywords of new objects to the right facets. For example, for a new image on Flickr that has the user-provided tags “sheep”, “fox”, “mountain”, and “fields”, our classifier will put the words “sheep” and “fox” under the “Animals” facet, while the words “mountain” and “fields” go under “Topographic Features”, even though the classifier may not have encountered some of the keywords beforehand. Second, for the collections with free-text objects such as The New York Time archive, we present an *unsupervised* technique that fully automates the extraction of useful facets from free-text objects (Sect. 7.2.3). In particular, we observe, through a pilot study, that facet terms rarely appear in text documents, which implies that we need external resources to identify useful facet terms. For this, we first identify important phrases in each document. Then, we expand each phrase with “context” phrases using external resources, such as WordNet [100] and Wikipedia,¹² causing facet terms to appear in the expanded collection. Finally, we compare the term distributions in the original collection and the expanded collection to identify the terms that can be used to construct browsing facets. Our extensive user studies, using the Amazon Mechanical Turk service, show that our techniques produce facets with high precision and recall, superior to existing approaches, and help users locate interesting objects fast.

7.2.2 Supervised Facet Extraction for Collections of Text-Annotated Items

Now, we describe our approach for extracting useful facets when we have access to some descriptive user-provided annotation, such as in the Corbis collection or on YouTube. One of the potential problems for constructing a concept hierarchy for

¹²<http://www.wikipedia.org>.

such a collection is that the same collection can be browsed in many different, orthogonal ways. Consider, for example, how a user can browse the schedule of TV programs. It is possible to browse by the time facet, by the TV channel facet, or by the title facet. It is also possible to browse by the actor facet, or by many other facets. Mixing facet-specific terms from multiple facets while constructing a single hierarchy can result in an awkward hierarchy. For example, an actor might be classified under the term “Monday” because he/she appears on a sitcom that is aired every Monday night, and therefore, the hierarchy would have the parent–child relation that assigns the actor’s name under the node “Monday” in the hierarchy. While it might be perfectly valid to assume this relation based on the co-occurrence of the term “Monday” and the name of the actor across the collection, this is not a structure that is useful for browsing. This type of relation contributes to the awkwardness of the resulting hierarchy. In short, having the items of a collection associated with descriptive keywords, such as with YouTube video clips, is not sufficient to produce useful hierarchies. Organizing these keywords across facets is a key step before proceeding to hierarchy construction.

While collections such as YouTube and Flickr lack such organization, a set of items in the Corbis collection has its keywords organized across predefined facets. For example, according to this Corbis set the words “cat” and “dog” are under the “Animals” facet, while the words “mountain” and “fields” are under “Topographic Features”. To be able to organize the keywords of a new item across the Corbis predefined facets, we use the data in this set to train a machine learning algorithm to classify keywords in the appropriate facets. In our approach, we treat the facet as a target classification class and the keywords as classification features. Unfortunately, such a straightforward approach does not generalize. A classifier trained in this way will correctly classify only words that have been assigned to facets before. A classifier might correctly classify the words “cat” and “dog” in the “Animals” facet, but a new word, such as “sheep”, that was not among the keywords of the training data will not be assigned to any facet.

To allow our technique to generalize for unseen keywords, we rely on the observation that keywords under different facets tend to have different “hypernyms”.¹³ Based on this observation, we expand each keyword using its hypernyms from a lexical corpus, such as WordNet. After the expansion, each keyword is represented as a set of words. For example, the word “cat” is represented as “cat, feline, carnivore, mammal, animal, living being, object, entity”. The new representation allows the classifier to generalize more easily and assign unseen words to the correct facets.

However, using hypernyms does not resolve the problem of sense disambiguation. Each word can have different meanings according to its context. Consider the word “kid”, which can mean either a young person or a young goat. Before assigning this word to a facet, we have to first decide the intended meaning of the word. To identify the correct meaning, we exploit the fact that keywords are associated with objects and each object is characterized by a set of other keywords, which provide

¹³A hypernym is a linguistic term for a word whose meaning includes the meanings of other words, as the meaning of vehicle includes the meaning of car, truck, motorcycle, and so on.

valuable clues for the writer’s intended meaning of the word. (The use of context is the basis of many techniques [178] for sense disambiguation.¹⁴) For example, when the word “kid” appears together with the words “goat” and “grazing”, then “kid” is much more likely to refer to a young goat than to a child.

Based on the observations noted above, we treat facet classification as a text classification problem. In text classification [93, 168], we characterize each document using a set of words; based on the presence of these words across categories, we train a classifier to assign documents to the appropriate categories. In our case, we treat each keyword as *three sets* of words. The first set of words contains the keyword itself, the second set contains the hypernyms of all the senses of the keyword, and the third set contains the other keywords associated with the object.

Specifically, our algorithm for assigning keywords to facets performs the following steps:

1. Obtain a collection D of text-annotated objects. Each object $d_i \in D$ has a set of associated keywords k_{i1}, \dots, k_{in} and each keyword k_{ij} is assigned to a facet F_{ij} .
2. For each keyword–facet pair $k_{ij}–F_{ij}$:
 - (a) Define the facet F_{ij} as the target class.
 - (b) Add the keyword k_{ij} in the first set of words.
 - (c) Add the hypernyms of k_{ij} in the second set of words.
 - (d) Add the other keywords associated with d_i (and their hypernyms) in the third set of words.
3. Train a document classifier over the prepared training data.

After training the classifier, we can use it over a new set of annotated objects to identify the facets that appear in the collection. After running the classifier over the keywords of the new objects, we can examine which facets appear frequently in the new data and use these facets for browsing. Empirically, we observed that facets that appear in 5% of the data can be useful for locating content of interest. We gathered our training data from a set of annotated images from the Corbis collection, which contained a comprehensive set of facets.¹⁵ We describe the experimental settings and report the results in Sect. 7.2.4. One “disadvantage” of supervised learning techniques is that they cannot “discover” new types of facets. In the next section, we describe an approach to identify new, previously unknown dimensions for browsing.

¹⁴We should emphasize that disambiguation for facet extraction is easier than the general problem of sense disambiguation. First, the context keywords are of high quality, something that is not always the case in natural language sentences. Second, and most importantly, while a word might have multiple senses, the senses are often closely related (see, for example, the WordNet senses for “gear” and “battle”). While sense disambiguation is hard for such words, closely related senses typically correspond to the same facet (“Generic Thing” for “gear” and “Action, Process, or Activity” for “battle”), eliminating the need for disambiguation for facet extraction.

¹⁵The whole collection contains more than 3 million images and 38 facets.

7.2.3 *Unsupervised Facet Extraction for Collections of Text Documents*

So far, the identification of the facets was either a manual procedure, or relied on a priori knowledge of the facets that can potentially appear in the underlying collection. Now, we describe our approach for extracting useful facets when descriptive keywords are not available for the collection items. In particular, we focus on the important family of collection of text documents. A key characteristic of documents in such collections is that they contain a relatively large number of words (and, correspondingly, of potential facets for interaction). This is in contrast to the text-annotated collections of the previous section, where each item in the collection generally has a much lower number of (often highly descriptive) keywords in the user-provided annotations, and a portion of the items, as in Corbis, has its keywords organized across a predefined set of facets.

To examine the largest hurdles for generating faceted hierarchies on top of news collections, we ran a small pilot study (Sect. 7.2.3.1) to examine what navigational structures would be useful for people who are browsing a news archive and to find clues on how to discover useful facets accompanied with descriptive keywords for each text document in Newsblaster. Our conclusions from the pilot study helped shape our approach for extracting useful facets and descriptive keywords from news articles (Sect. 7.2.3.2). Our approach assumes that high-level facet terms rarely appear in the documents. For example, consider the named entity “*Jacques Chirac*”. This term would appear under the facet “*People* → *Political Leaders*”. Furthermore, this named entity also implies that the document can be potentially classified under the facet “*Regional* → *Europe* → *France*”. Unfortunately, these (facet) terms are not guaranteed to appear in the original text document. However, if we expand the named entity “*Jacques Chirac*” using an external resource, such as Wikipedia, we can expect to encounter these important *context terms* with greater frequency. Our hypothesis is that facet terms emerge after the expansion, and their frequency rank increases in the new, expanded collection. In particular, we take advantage of this property of facet terms to automatically discover, in an unsupervised manner, a set of candidate facet terms from the expanded news articles. We then automatically group together facet terms that belong to the same facet using a hierarchy construction algorithm [274] and build the appropriate browsing structure for each facet using our algorithm for the construction of faceted interfaces.

7.2.3.1 A Pilot User Study

For our initial pilot study, we recruited 12 students studying either journalism or art history. We randomly chose a thousand stories from *The New York Times* archive, and we asked the student annotators to manually assign each story to several facets that they considered appropriate and useful for browsing. The most common facets

Table 7.5 Facets identified by human annotators in a small collection of 1,000 news articles from The New York Times

Facets
Location
Institutes
History
People
↔ Leaders
Social Phenomenon
Markets
↔ Corporations
Nature
Event

identified by the annotators were “*Location*”, “*Institutes*”, “*History*”, “*People*”, “*Social Phenomenon*”, “*Markets*”, “*Nature*”, and “*Event*”. For these facets, the annotators also identified other “sub-facets” such as “*Leaders*” under “*People*” and “*Corporations*” under “*Markets*”.

From the results of the pilot study, we observed that the terms for the useful facets do not usually appear in the news stories. (In our study, this phenomenon surfaced in 65% of the user-identified facet terms.) Typically, journalists do not use general terms, such as those used to describe facets, in their stories. For example, a journalist writing a story about *Jacques Chirac* will not necessarily use the term “*Political Leader*” or the term “*Europe*” or even “*France*”. Such (missing) *context terms* are useful for identifying the appropriate facets for the story.

This pilot experiment demonstrated that a tool for the automatic discovery of useful facet terms should exploit external resources that could return the appropriate facet terms. Such an external resource should provide the appropriate context for each of the terms that we extract from the collection. As a result, a key step of our approach corresponds to an expansion procedure, in which the *important terms* from each news story are expanded with *context terms* derived from external resources. The expanded documents then contain many of the terms that can be used as facets. Next, we describe our algorithm in detail, showing how to identify these *important* and *context* terms.

7.2.3.2 Automatic Facet Discovery

The results of our pilot study from Sect. 7.2.3.1 indicate that general facet terms rarely occur in news articles. To annotate a given story with a set of facets, we normally skim through the story to identify important terms and associate these terms with other more general terms, based on our accumulated knowledge. For example, if we conclude that the phrase “*Steve Jobs*” is an important aspect of a news story, we can associate this story with general terms such as “*personal computer*”, “*en-*

Input: Original collection D , term extractors E_1, \dots, E_k

Output: Annotated collection $I(D)$

Step 1: Extract *all* terms from each document d in collection D and compute for each term t its term frequency $Freq_O(t)$.

Step 2: Execute *all* extractors E_1, \dots, E_k on each document d in collection D to identify d 's important terms $E_i(d)$ based on extractor E_i , and compute $I(d)$ to be the union $E_1(d) \cup \dots \cup E_k(d)$.

Algorithm 7.1: Identifying important terms within each document

entertainment industry”, or “technology leaders”. Our techniques operate in a similar way. In particular, our algorithm follows these steps:

1. For each document in the collection, identify the *important* terms *within* the document that are useful for characterizing the contents of the document.
2. For each important term in the original document, query one or more external resources and retrieve the *context* terms that appear in the results. Add the retrieved terms to the original document, in order to create an expanded, “context-aware” document.
3. Analyze the frequency of the terms, both in the original collection and the expanded collection, and identify the candidate facet terms.

Identifying Important Terms The first step of our approach (see Algorithm 7.1) identifies important terms¹⁶ in the text of each document. We consider the terms that carry information about the different aspects of a document to be important. For example, consider a document d that discusses the actions of *Jacques Chirac* during the *2005 G8 summit*. In this case, the set of important terms $I(d)$ may contain two terms, as follows:

$$I(d) = \{\textit{Jacques Chirac}, \textit{2005 G8 summit}\}$$

We use the next three techniques in this term selection step:

- **Named Entities (LPNE):** We use a named-entity tagger to identify terms that provide important clues about the topic of the document. Our choice is reinforced by existing research (e.g., [107, 129]) that shows that the use of named entities increases the quality of clustering and of news event detection. We build on these ideas and use the named entities extracted from each news story as important terms that capture the important aspects of the document. In our work, we use the named-entity tagger provided by the LingPipe¹⁷ toolkit.
- **Yahoo Terms (YTERM):** We use the “Yahoo Term Extraction”¹⁸ web service, which takes as input a text document and returns a list of significant words or

¹⁶By *term*, we mean single words and multi-word phrases.

¹⁷<http://www.alias-i.com/lingpipe/>.

¹⁸<http://developer.yahoo.com/>.

phrases extracted from the document.¹⁹ We use this service as a second tool for identifying important terms in the document.

- **Wikipedia Terms (WTERM):** We developed our own tool to identify important aspects of a document based on Wikipedia entities. Our tool is based on the idea that an entity is typically described in its own Wikipedia page. To implement the tool, we downloaded the contents of Wikipedia and built a relational database that contains (among other things) the titles of all the Wikipedia pages. Whenever a term in a document matches a title of a Wikipedia entry, we mark the term as important. If there are multiple candidate titles, we pick the longest title to identify the important term.

Furthermore, we exploit the link structure of Wikipedia to improve the detection of important terms. First, we exploit the “*redirect*” pages, to improve the coverage of the extractor. For example, the entries “Hillary Clinton”, “Hillary R. Clinton”, “Clinton, Hillary Rodham”, “Hillary Diane Rodham Clinton”, and others redirect to the page with title “Hillary Rodham Clinton”. By exploiting the redirect pages, we can capture multiple variations of the same term, even if the term does not appear in the document in the same format as in the Wikipedia page title. (We will also use this characteristic in Step 2, to derive context terms.) In a similar manner, we also exploit the *anchor text* from other Wikipedia entries to find different descriptions of the same concept. Even though the anchor text has been used extensively in the web context [50], we observed that the anchor text works even better within Wikipedia, where each page has a specific topic.

Beyond the three techniques described above, we can also follow alternative approaches in order to identify important terms. For instance, we can use domain-specific vocabularies and ontologies (e.g., from the *Taxonomy Warehouse*²⁰ by Dow Jones) to identify important terms for a domain. Here, due to the lack of appropriate text collections that could benefit from such resources, we do not consider this alternative. Still, we believe that exploiting domain-specific resources for identifying important terms can be useful in practice.

The next step of the algorithm uses important document terms to identify additional context terms, relevant to the documents.

Deriving Context Using External Resources In Step 2 of our approach, we use the identified important terms to expand each document with relevant context (see Algorithm 7.2). As we discussed in Sect. 7.2.3.1, in order to build facets for browsing a text collection, we need more terms than the ones that appear in the collection. To discover the additional terms, we use a set of *external resources* that can provide the additional context terms when queried appropriately.

For example, assume that we use Wikipedia as the external resource, trying to extract context terms for a document d with a set of important terms

¹⁹We have observed empirically that the quality of the returned terms is high. Unfortunately, we could not locate any documentation about the internal design of the web service.

²⁰<http://www.taxonomywarehouse.com/>.

Input: Annotated collection $I(D)$, external resources R_1, \dots, R_m

Output: Contextualized collection $C(D)$

Step 1: Query each external resource R_i to retrieve the context terms $R_i(t)$ for each important term t in $I(d)$ of each document d .

Step 2: Create for each document d the context terms $C(d)$ as the union of all context terms $R_i(t)$ of all terms t in $I(d)$ and all external resources R_1, \dots, R_m .

Step 3: Augment document d with context terms $C(d)$.

Algorithm 7.2: Deriving context terms using external resources (Sect. 7.2.3.2)

$I(d) = \{\textit{Jacques Chirac, 2005 G8 summit}\}$. We query Wikipedia with the two terms in $I(d)$, and we analyze the returned results. From the documents returned by Wikipedia, we identify additional context terms for the two terms in the original $I(d)$: the term *President of France* for the original term *Jacques Chirac* and the terms *Africa debt cancellation* and *global warming* for the original term *2005 G8 summit*. Therefore, the set $C(d)$ contains three additional context terms, namely, *president of France*, *Africa debt cancellation*, and *global warming*.

In our work, we use four external resources, and our framework can be naturally expanded to use more resources, if necessary. We used two existing applications (WordNet and Google) that have proved useful in the past and developed two new resources (Wikipedia Graph and Wikipedia Synonyms). Specifically, the resources that we use are the following:

- **Google (GOOG):** The web can be used to identify terms that tend to co-occur frequently. Therefore, as one of the expansion strategies, we query Google with a given term, and then retrieve as context terms the most frequent words and phrases that appear in the returned snippets.
- **WordNet Hypernyms (WORDNET):** Previous studies in the area of automatic generation of facet hierarchies [79, 282] observed that WordNet *hypernyms* are good terms for building facet hierarchies. Based on our previous experience [79], hypernyms are useful and high-precision terms, but they tend to have low recall, especially when dealing with named entities (e.g., names of politicians) and noun phrases (e.g., “due diligence”). Therefore, WordNet should not be the only resource used but should be complemented with additional resources. We discuss such resources next.
- **Wikipedia Graph (WGRAPH):** A useful resource for discovering context terms is Wikipedia. In particular, the links that appear in the page of each Wikipedia entry can offer valuable clues about associations with other entries. To measure the level of association between two Wikipedia entries t_1 and t_2 that are connected with a link $t_1 \rightarrow t_2$, we examine two values: the number of outgoing links $out(t_1)$ from t_1 to other entries and the number of incoming links $in(t_2)$ pointing to t_2 from other entries. Using *tf.idf*-style scoring, we set the level of association to $\log(N/in(t_2))/out(t_1)$, where N is the total number of Wikipedia entries. (Notice that the association metric is not symmetric.) When querying the “Wikipedia

Graph” resource with a term t , the resource returns the top- k terms²¹ with the highest scores. For example, there is a page dedicated to the Japanese samurai “Hasekura Tsunenaga”. The “Hasekura Tsunenaga” page is linked to the pages “Japanese Language”, “Japanese”, “Samurai”, “Japan”, and several other pages. There are more than 6 million entries and 35 million links in the Wikipedia graph, creating an informative graph for deriving context. As expected, the derived context terms will be both more general and more specific terms. We will examine in Sect. 7.2.3.2 how we identify the more general terms, using statistical analysis of the term frequencies in the original collection and in the contextualized collection.

- **Wikipedia Synonyms (WSYNONYMS):** We constructed Wikipedia Synonyms as a resource that returns variations of the same term. As we described earlier, we can use the Wikipedia redirect pages to identify variations of the same term. To achieve this, we first group together the titles of entries that redirect to a particular Wikipedia entry. For example, the entries “Hillary Clinton”, “Hillary R. Clinton”, “Clinton, Hillary Rodham”, and “Hillary Rodham Clinton” are considered synonyms since they all redirect to “Hillary Rodham Clinton”.

Although redirect pages return synonyms with high accuracy, there are still variations of a name that cannot be captured like this. For such cases, we use the anchor text that is being used in other Wikipedia pages to link to a particular entry. For example, there is a page dedicated to the Japanese samurai “Hasekura Tsunenaga”. The “Hasekura Tsunenaga” has also pointers that use the anchor text “Samurai Tsunenaga”, which can also be used as a synonym. Since anchor text is inherently noisier than redirects, we use a form of *tf.idf* scoring to rank the anchor text phrases. Specifically, the score for the anchor text p pointing to a Wikipedia entry t is $s(p, t) = tf(p, t)/f(p)$, where $tf(p, t)$ is the number of times that the anchor phrase p is used to point to the Wikipedia entry t , and $f(p)$ is the number of different Wikipedia entries pointed to by the same text p .

At the end of Step 2, we create a *contextualized* collection in which each document contains both the original terms and a set of context terms. Next, we describe how we can use the term frequencies in the original and in the contextualized collection to identify useful facet terms.

Comparative Term Frequency Analysis So far, we have identified important terms in each document and used them to expand the document with general relevant context for each document. In this section we describe how we process both the expanded and original collections to identify terms that are good candidates for facet terms.

Our algorithm is based on the intuition that facet terms are infrequent in the original collection but frequent in the expanded one. So, to identify such terms, we need first to identify terms that occur “more frequently” and then make sure that this difference in frequency is statistically significant, and not simply the result of noise. To measure the difference in frequency, we define the next two functions:

²¹We set $k = 50$ in our experiments.

Input: Original collection D , contextualized collection $C(D)$

Output: Useful facet terms $Facet(D)$

Step 1: Compute $df(t)$ for each term t in collection D as the total number of documents that contain t .

Step 2: Compute $df_C(t)$ for each term t in collection $C(D)$ as the total number of documents that contain t .

Step 3: Let $df(t)$ be equal to zero if term t occurs in collection $C(D)$ but does not occur in collection D .

Step 4: Compute the functions $Shift_f(t)$ (equation (7.1)) and $Shift_r(t)$ (equation (7.4)) for each term t in collection $C(D)$, and add t to the facet terms $Facet(D)$ if both functions are positive.

Step 5: Sort $Facet(D)$ in increasing order of $-\log \lambda_t$ (equation (7.5)) and return the top- k terms.

Algorithm 7.3: Identifying important facet terms by comparing the term distributions in the original and in the contextualized collection (Sect. 7.2.3.2)

- **Frequency-Based Shifting:** For each term t , we compute the frequency difference as:

$$Shift_f(t) = df_C(t) - df(t) \quad (7.1)$$

where $df_C(t)$ and $df(t)$ are the frequencies of term t in the contextualized collection and the original collection, respectively. Due to the Zipfian nature of the term frequency distribution [337], this function tends to favor terms that already have a high frequency in the original collection. High-frequency terms demonstrate higher increases in frequency, even if they are less popular in the expanded collection compared to the original one. The inverse problem appears if we use ratios instead of differences. To avoid the shortcomings of this approach, we introduce a rank-based metric that measures the differences in the ranking of the terms.

- **Rank-Based Shifting:** We use a function B that assigns terms to bins based on their ranking in the original and the contextualized collections, as follows:

$$B(t) = \lceil \log_2(Rank(t)) \rceil \quad (7.2)$$

$$B_C(t) = \lceil \log_2(Rank_C(t)) \rceil \quad (7.3)$$

where $Rank(t)$ is the rank of the term t in the original collection, and $Rank_C(t)$ is the rank of the term t in the contextualized collection. After computing the bin $B(t)$ and $B_C(t)$ of each term t , we define the shifting function as follows:

$$Shift_r(t) = B(t) - B_C(t) \quad (7.4)$$

In our approach, a term becomes a candidate facet term only if both $Shift_f(t)$ and $Shift_r(t)$ are positive. After identifying terms that occur more frequently in the con-

textualized collection, the next test verifies that the difference in frequency is statistically significant. A test such as the chi-square test [65] could be potentially used to identify important frequency differences. However, due to the power-law distribution of the term frequencies [337], many of the underlying assumptions for the chi-square test do not hold for text frequency analysis [96]. Therefore, we use the *log-likelihood* statistic, assuming that the frequency of each term in the (original and contextualized) collections is generated by a binomial distribution:

- **Log-Likelihood Statistic:** For a term t with document frequency df in the original collection D and frequency df_C in the contextualized collection $C(D)$, the log-likelihood statistic for the binomial case is:

$$-\log \lambda_t = \log L(p_1, df_C, |D|) + \log L(p_2, df, |D|) - \log L(p, df, |D|) - \log L(p, df_C, |D|) \quad (7.5)$$

where $\log L(p, k, n) = k \log(p) + (n - k) \log(1 - p)$, $p_1 = \frac{df_C}{|D|}$, $p_2 = \frac{df}{|D|}$, and $p = \frac{p_1 + p_2}{2}$. For an excellent description of the log-likelihood statistic see the seminal paper by Dunning on the subject [96].

The shift functions and the log-likelihood test return a set of terms $Facet(D)$ that can be used for faceted navigation (see Algorithm 7.3). Once we have identified these terms, it is relatively easy to build the actual hierarchies. For our work, we used the subsumption algorithm by Sanderson and Croft [260], with satisfactory results, although newer algorithms [274] may improve performance further.

7.2.4 Evaluating Our Supervised Facet Extraction Technique

In this section, we present the experimental evaluation of our supervised extraction technique of Sect. 7.2.2. First, we describe our experimental settings in Sect. 7.2.4.1, then we evaluate our facet extraction technique in Sect. 7.2.4.2.

7.2.4.1 Experimental Settings

Our classifier variants are trained and tested over as a set of keywords associated across a predefined set of facets. We now describe this data set in and then, we briefly describe our classifier variants. Finally, we present the evaluation metrics that we use to compare our classifier variants.

Data Collection To evaluate our classifier of Sect. 7.2.2, we need a collection of items assigned to descriptive keywords across a set of predefined facet terms. For our experiments, we use 36,820 annotated images from the Corbis image collection, which we mentioned in Sect. 7.2.1. Each image has a title and free-text description, and is associated with a set of keywords. Each keyword is assigned manually by the

Table 7.6 List of the 14 most commonly used facets in Corbis

Facet	Description
ABC	Abstract Concepts
APA	Action, Process, or Activity
ATT	Attributes
ATY	Anatomy
GAN	Generic Animal
GCF	Generic Cultural Features and Works
GEV	Generic Event
GPL	Generic Plant
GTF	Generic Topographic Feature
GTH	Generic Thing
NCF	Named Cultural Features and Works
NORG	Named Organizations and Groups
NTF	Named Topographic Feature
RPS	Religious, Political, Philosophical, and Social Issues

Corbis annotators to one of the 38 facets that are used by Corbis. In total there are 65,521 unique keywords, primarily assigned to 14 of the 38 facets. The remaining 24 facets had less than 100 keywords assigned to them, so we ignored them for the purposes of our evaluation. Table 7.6 lists the 14 most commonly used facets with their full names. Since our facet extraction algorithm relies on the existence of pre-annotated data, we picked 11,000 keywords and their associated facets to train and test our algorithm. To avoid any bias, we randomly picked the 11,000 keywords from 11,000 randomly selected images, choosing one keyword per image.

Techniques for Comparison We evaluated three versions of our classifier using Support Vector Machines (SVM) with linear kernels. The first classifier, which serves as a weak baseline, does not use WordNet hypernyms or the associated keywords as features. The second, which serves as a strong baseline, uses WordNet hypernyms. The last classifier, which serves as our strongest version, uses both WordNet hypernyms and the associated keywords as additional features.

Evaluation Metrics To compare the three versions of our classifier, we use the following metrics, which are commonly used in statistical classification:

Precision: The precision of a classification class c is defined as the number of keywords (i.e., learning examples) that are truly labeled (by humans) and automatically classified (by a classifier) as c divided by the total number of keywords classified as c .

Recall: The recall of a classification class c is defined as the number of keywords that are both truly labeled and automatically classified as c divided by the total number of keywords that are truly labeled as c .

Table 7.7 The average performance of the facet extraction technique (strong classifier) for each of the 14 facets in the Corbis data set. Results are obtained using 10-fold cross-validation. (Table 7.6 contains the full names of the facets)

Facet	Precision	Recall	F_1 -measure
ABC	85.20%	87.60%	86.38%
APA	75.80%	75.80%	75.80%
ATT	78.20%	83.50%	80.76%
ATY	80.00%	81.30%	80.64%
GAN	92.90%	92.90%	92.90%
GCF	74.70%	76.76%	75.72%
GEV	79.40%	56.30%	65.88%
GPL	81.70%	90.10%	85.69%
GTF	86.70%	75.00%	80.43%
GTH	87.70%	83.00%	85.29%
NCF	82.40%	87.57%	84.91%
NORG	75.40%	76.58%	75.99%
NTF	82.40%	80.30%	81.34%
RPS	85.60%	76.30%	80.68%
Average	82.01%	80.22%	80.89%

F_1 -measure: The F_1 -measure of a classification class is defined as the weighted harmonic mean of precision and recall, or

$$\frac{2 \cdot \textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

7.2.4.2 Experimental Results

Initially, we tested the accuracy of the weak baseline and, as expected, the classifier could not generalize to unseen examples. Its accuracy, as measured by the average F_1 -measure, was 10%, only slightly higher than the accuracy of a random classifier. By adding the WordNet hypernyms, the performance of the baseline classifier improved considerably, reaching an average F_1 -measure of 71%. This improvement confirmed our hypothesis that hypernyms are useful features for allocating keywords to facets. Nonetheless, the sense ambiguity is still a problem in this case: after adding the remaining keywords from each document as extra features, performance improved considerably. Our strong classifier reached an average F_1 -measure of 81% (see Table 7.7).

We also compared our strong classifier against variations of other techniques. One hypothesis was that we can create facets by picking some high-level hypernyms from WordNet, which can serve as root nodes for the corresponding facets.

For example, the term “animal/fauna” in WordNet could serve as the root node for the “Animal” facet. Subsequently, all terms that have “animal/fauna” as a hypernym could be assigned to the “Animal” facet. (This approach is close in spirit to the hierarchy construction algorithm in [281].) To test the accuracy of this approach, we trained RIPPER [71], a *rule-based* classifier, using the keywords and their hypernyms as features. The average F_1 -measure in that case was close to 55%, significantly worse than the corresponding results for SVMs. The results also highlighted that some classes (facets) work well with simple, rule-based assignments of terms to facets, but there are other classes that need more elaborate classifiers. For example, for the facet GAN (**G**eneric **A**Nimals) the rule-based classifier resulted in an F_1 -measure of 93.3%, showing that simple rules work well for this facet. However, for the APA facet (**A**ction, **P**rocess, or **A**ctivity) the F_1 -measure was only 35.9%, showing that simple rules do not work well for such a complex facet.

7.2.5 Evaluating Our Unsupervised Facet Extraction Technique

In this section, we discuss the experimental evaluation of our unsupervised extraction technique of Sect. 7.2.3. First, we discuss the settings we used for our evaluation, including the data sets and human judgments, our technique variations, and our evaluation metrics. Then, we describe how we evaluated the recall and precision of our techniques. Finally, we present our results on the efficiency of our techniques, and finally we briefly discuss some results of a user study that demonstrates the usefulness of the derived faceted navigational structures.

7.2.5.1 Experimental Settings

Our technique receives as input a set of free-text documents and produces as output a set of hierarchically structured facets that can be used to browse the text collection in an OLAP-like manner. Sect. 7.2.5.1 describes three data collections that we use to evaluate the variants of our techniques, and an Amazon Mechanical Turk study to evaluate the quality of the extracted facets by human annotators. Section 7.2.5.1 describes the variants of our technique. Finally, Sect. 7.2.5.1 presents the evaluation metrics for the experiments.

Data Collections

Single Day of The New York Times (SNYT): A collection of 1,000 news stories from The New York Times archive, covering one day of news from November 2005.

Single Day of Newsblaster (SNB): A collection of 17,000 news stories retrieved by Newsblaster [183] from 24 news sources on one day in November 2005. We use this data set to test how our techniques work over data from multiple sources.

Month of The New York Times (MNYT): A collection of 30,000 news stories from The New York Times archive, covering one month of news from November 2005.

Human Extracted Facet Terms: Since our experiments required extensive input from users and there is no standard benchmark for evaluating the quality of the generated facets, we conducted a human study on the Amazon Mechanical Turk service. Specifically, in our study each Mechanical Turk annotator had to read a story and identify terms that can be used for faceted navigation. We informed the annotators that the terms may or may not appear in the document, and it was up to the annotator to determine whether the terms were useful for the task of faceted navigation. For each article, the annotators were asked to provide up to 10 candidate facet terms. We instructed them to choose terms that were clearly defined, mutually exclusive, and covered as many aspects, properties, or characteristics of the story as possible. Each of the 1,000 stories in *SNYT* was examined by five annotators. For *SNB* and *MNYT*, we picked a random sample of 1,000 stories, and again each story was annotated by five annotators. To eliminate annotation errors or idiosyncratic annotations, we considered an annotation to be valid if at least two of the annotators used the same facet term for a given story. The final set contained 633 facet terms for *SNYT*, 756 facet terms for *SNB*, and 703 terms for *MNYT*. This indicates that the number of facet terms increases relatively slowly as the number of news sources rises (i.e., from *SNYT* to *SNB*) and as we expand the examined time period (i.e., from *SNYT* to *MNYT*). To make sure that this is not an artifact of our sampling approach, we also performed a sensitivity test, examining how the number of facet terms increases as we increase the number of stories in each data set from 100 to 1,000. At 100 documents, we discovered approximately 40% of the facet terms, and at 500 documents we discovered approximately 80% of the facet terms for each of the data sets. Therefore, we believe that the marginal increase in facet terms if we annotated all 17,000 articles for *SNB* and all 30,000 articles for *MNYT* would be relatively small. Figure 7.3 shows a sample of the most frequently identified facet terms for the three data sets.

Precise Extracted Facet Terms: Our techniques extract a significant number of terms that the Mechanical Turk annotators did not identify when marking the important facet terms in the documents (see earlier details on the study). Still, when examining the extracted faceted terms with the generated hierarchies, we could easily determine whether a particular facet term accurately depicts the contents of the underlying collection. So, we asked the annotators to examine the extracted facet terms with the generated hierarchies and determine for each facet term the following: (a) whether the term is useful in the facet hierarchy and (b) whether the term is accurately placed in the hierarchy. To ensure the quality of the annotations, the Mechanical Turk annotators that participated in this experiment had to pass a *qualification test*. To conduct our test, we initially picked random subtrees of the Open Directory²² hierarchy as our “correct” hierarchies.

²²<http://www.dmoz.org>.

Table 7.8 A list of our external resources (Sect. 7.2.3.2)

Name	Description
GOOG	Querying Google and using the results snippets.
WORDNET	Querying WordNet for related Hypernyms.
WSYNONYMS	Querying Wikipedia for related Synonyms.
WGRAPH	Querying Wikipedia Graph for related aspects.
<i>All</i>	Combining all resources.

Table 7.9 A list of our term extractors techniques

Name	Description
LPNE	Extracting terms using LingPipe’s name entity tagger.
YTERM	Extracting terms using “Yahoo Term Extraction” web service.
WTERM	Extracting terms using our Wikipedia term extractor.
<i>All</i>	Combining all extractors.

To generate “noisy” hierarchies, we randomly perturbed some parent–child relations and randomly swapped terms across separate hierarchy subtrees. Then, during the qualification test, each prospective Mechanical Turk annotator had to annotate 20 correct and incorrect hierarchies and was only allowed to proceed with the real annotation task if he or she gave the correct answer for at least 18 out of 20 hierarchies. As in the case of the earlier study, each facet term was examined by five Mechanical Turk annotators. We only consider a term to be a precise facet term if at least four annotators marked the facet term as precise.

Techniques for Comparison We create several variations of the general technique that we described in Sect. 7.2.3 based on (1) four external resources we use to expand the collection, namely, GOOG, WORDNET, WSYNONYMS, and WGRAPH; and (2) three term extractor techniques, namely, LPNE, YTERM, and WTERM. Tables 7.8 and 7.9 list the details of our external resources and term extractors, respectively.

Evaluation Metrics To evaluate the different variations of our technique, we use the following metrics.

Recall: This metric measures how many of the manually extracted facet terms were also identified by our techniques. We define recall as the fraction of the manually extracted facet terms that were also extracted by our techniques.

Precision: We also evaluate the precision of the extracted facets using the same methodology that we used for estimating recall. However, our techniques extract a significant number of concepts that the Mechanical Turk annotators did

politics, money, market, government, history, competition, people, education, location, new york, power, terrorism, war, baseball, event, biography, business, children, development, health, music, real estate, sports, change, comeback, crime, entertainment, greed, national security, nature poverty, spending, success, pride, technology, winning, anger, architecture, branding, foreign lands, bush administration, campaign, capitalism, challenges, civil unrest, civil war, community, compromise, computers, consumer confidence, corruption, countries, culture of fear, disagreement, distribution, power of the Internet, expectations fear, humor, innovation, investigation, Iraq, Italian culture, jobs, leadership, moving, opportunity, optimism, planning, players, police, public relations, publicity, religion, religious, warfare rights, statistics, support, time, torture, U.S., violence, wealth, youth

Fig. 7.3 A sample of the most frequently identified facet terms, as extracted by human annotators. All the terms above were anonymously selected by at least two annotators

year, new, time, people, state, work, school, home, mr, report, game, million, week, percent, help, right, plan, house, high, world, american, month, live, call, thing

Fig. 7.4 Facet terms identified by a simple subsumption-based algorithm [260], without using our techniques

not identify when marking the important facet terms in the documents. We consider an extracted term to be a “precise” facet term if at least four annotators marked it as precise. The precision is then the ratio of precise extracted terms over the total number of extracted terms.

Efficiency: We also measure another important aspect of our techniques, namely, the time required to extract useful facet terms for a collection.

7.2.5.2 Experimental Results

Recall Our first step was to measure how many of the manually extracted facet terms were also identified by our techniques. To examine the individual effect of each term extractor and of each external resource, we computed the fraction of identified facet terms for each of the possible combinations of term extractor and external resource. We also computed the recall for the case in which we used *all* the term extractors and *all* the external resources.

We list the results in Tables 7.10, 7.11, and 7.12 for the *SNYT*, *SNB*, and *MNYT* data sets, respectively. The results were consistent across data sets. In general, recall increases as we increase the number of term extractors and as we increase the number of external resources. *WSYNONYMS* and *WORDNET* tend to perform relatively poorly compared to *Google* and *WGRAPH*, especially when using named entities (*LPNE*) as the term extractor. However, both resources are helpful when combined with *GOOG* and *WGRAPH*, and increase the overall recall of the results.

Precision Recall that we consider a facet term to be “precise” if and only if it was identified by our annotator as useful within the generated hierarchy of the facet

Table 7.10 The recall of the extracted facets, as judged by the human annotators for the *SNYT* data set

External resource	Term extractors			
	LPNE	YTERM	WTERM	All
GOOG	0.529	0.703	0.761	0.819
WORDNET	0.090	0.510	0.491	0.592
WSYNONYMS	0.105	0.156	0.345	0.408
WGRAPH	0.632	0.791	0.801	0.881
All	0.746	0.891	0.899	0.945

Table 7.11 The recall of the extracted facets, as judged by the human annotators for the *SNB* data set

External resource	Term extractors			
	LPNE	YTERM	WTERM	All
GOOG	0.515	0.658	0.699	0.751
WORDNET	0.084	0.487	0.395	0.514
WSYNONYMS	0.112	0.162	0.306	0.314
WGRAPH	0.615	0.755	0.773	0.823
All	0.707	0.861	0.856	0.881

Table 7.12 The recall of the extracted facets, as judged by the human annotators for the *MNYT* data set

External resource	Term extractors			
	LPNE	YTERM	WTERM	All
GOOG	0.522	0.658	0.699	0.793
WORDNET	0.087	0.487	0.395	0.555
WSYNONYMS	0.109	0.146	0.331	0.392
WGRAPH	0.627	0.778	0.790	0.853
All	0.733	0.859	0.860	0.921

and is accurately placed in the hierarchy. And we defined the precision to be the ratio of precise terms over the total number of extracted facet terms. (We discuss the generation of faceted hierarchies in the following chapter.)

We list the precision results in Tables 7.13, 7.14, and 7.15 for the *SNYT*, *SNB*, and *MNYT* data sets, respectively. Again, the results were consistent across data sets. The highest precision hierarchies are those generated by WordNet; this is not surprising since the hypernyms naturally form a hierarchy. The use of Google as an external resource tends to lead to a drop in precision. In our implementation, for efficiency, we only use the terms that appear in the titles and snippets in the Google results; we do not retrieve the actual HTML pages of the returned results. This

Table 7.13 The precision of the extracted facets, as judged by the human annotators for the *SNYT* data set

External resource	Term extractors			
	LPNE	YTERM	WTERM	All
GOOG	0.615	0.769	0.751	0.678
WORDNET	0.923	0.901	0.932	0.915
WSYNONYMS	0.734	0.815	0.845	0.819
WGRAPH	0.828	0.813	0.842	0.827
All	0.817	0.796	0.858	0.866

Table 7.14 The precision of the extracted facets, as judged by the human annotators for the *SNB* data set

External resource	Term extractors			
	LPNE	YTERM	WTERM	All
GOOG	0.505	0.796	0.751	0.714
WORDNET	0.897	0.919	0.909	0.922
WSYNONYMS	0.633	0.904	0.875	0.853
WGRAPH	0.789	0.851	0.885	0.822
All	0.796	0.815	0.834	0.831

Table 7.15 The precision of the extracted facets, as judged by the human annotators for the *MNYT* data set

External resource	Term extractors			
	LPNE	YTERM	WTERM	All
GOOG	0.487	0.818	0.834	0.794
WORDNET	0.878	0.925	0.932	0.917
WSYNONYMS	0.691	0.851	0.880	0.879
WGRAPH	0.801	0.824	0.837	0.810
All	0.713	0.836	0.855	0.861

approach introduces a relatively large number of noisy terms. An interesting direction for future research is to examine whether the introduction of a term extraction mechanism from the HTML pages could improve the precision of our hierarchies. In contrast to Google, the use of the Wikipedia resources produces more precise hierarchies. Given the high precision of the WordNet- and Wikipedia-based hierarchies, it would be interesting to see if we could use ontologies that combine WordNet and Wikipedia in a single resource [283] as external resources.

Efficiency In our experiments, term extraction took 2–3 seconds per document, and the main bottleneck was the Yahoo! Term Extractor. If we eliminate the Ya-

hoo! Term Extractor, then we can process approximately 100 documents per second. Similarly, document expansion takes approximately one second per document when using Google as an external resource. Using Wikipedia and WordNet, which are stored locally, is significantly faster: we can process more than 100 documents per second, effectively making the term extraction the real bottleneck in the process. The facet term selection phase is extremely fast (i.e., about a few milliseconds), and we use an efficient hierarchy construction using the techniques described in [79], to create the facet hierarchies in 1–2 seconds.

In a real deployment scenario, we can considerably increase facet extraction efficiency by performing the term and context extraction offline. In this case, the results are ready before the real facet computation, which then takes only a few seconds and is almost independent of the collection size. (So, we can generate facet hierarchies over the complete collection *and* dynamically over a set of lengthy query results.) If term and context extraction need to be performed on-the-fly over thousands of documents, and it is important to compute the facet hierarchies quickly, then it would be preferable to avoid using web-based resources, such as Yahoo! Term Extractor and Google, and instead use only locally available resources, such as LingPipe, Wikipedia, and WordNet.

User Study Finally, we examine the reaction of users to automatically extracted facets. For this, we recruited five subjects to use our system to locate news items of interest, and we asked them to repeat the task 5 times. We provided a keyword-based search interface that was augmented with our faceted hierarchies located on the side. We measured how often during each search session the users clicked on the facet terms and how often they used the keyword-based search. We also measured the time required to finish the task. At the end of each session, we asked users to indicate their level of satisfaction, on a scale of 0–3, where 0 = dissatisfied, 1 = slightly dissatisfied, 2 = slightly satisfied, and 3 = satisfied.

We observed that, in the *first* interaction with the system, the users typed a keyword query of a named entity associated with the general topic in which they were interested (e.g., “war in Iraq”). Then they proceeded to locate news stories of interest by clicking on the facet hierarchies until they had generated a small subset of news stories associated with the same topic. Interestingly, in subsequent interactions with the system, the users started by using the faceted hierarchies directly, and their use of the keyword-search interface was gradually reduced by up to 50%. In addition, the time required to complete each task dropped by 25%, and the level of satisfaction remained statistically steady, with a mean level of 2.5 in the 0–3 scale. These results are consistent with previous research studies that relied on manually generated facet hierarchies [327] or on hierarchies extracted only from WordNet [282].

Results Summary The results of our user study indicate that users are generally comfortable using facet terms, and grow even more comfortable with greater exposure to them. Furthermore, by using the facets, users can locate items of interest more quickly, without any decline in satisfaction. The similarity of the interface with existing OLAP tools means that our techniques can be seamlessly integrated

with current OLAP systems that provide easy access to structured, relational data. Our techniques can expand existing OLAP systems to work over unstructured, or semi-structured data, allowing OLAP users to quickly discover interesting associations.

7.2.6 Further Discussion and Future Work

We evaluated our supervised approach over a text-annotated collection, refined with the use of training data in which descriptive keywords are assigned to the items in the collection and the keywords are organized across facets terms. We evaluated our unsupervised approach over a news collection with the benefit of a hypothesis that useful high-level browsing terms can be found in external knowledge resources. Unfortunately, we only evaluated the supervised approach over the Corbis collection mainly because we do not have access to other collections that are annotated in a similar manner as Corbis, and we only evaluated the unsupervised approach over collections of news articles. For future work, we are interested in combining the two approaches to extend our work for semi-annotated collections such as YouTube: each YouTube item (or video clip) is tagged with several keywords, as in the Corbis collection, but is not associated with useful facets, unlike in the Corbis collection.

Our unsupervised techniques rely on external informative resources and we only query three such resources. In fact, there are several other useful resources within specialized contexts that could be relatively straightforward to integrate into this framework. For instance, the *Taxonomy Warehouse*²³ developed by Dow Jones contains a large list of controlled vocabularies and specialized taxonomies that can be used for term identification and term expansion, respectively. For example, when browsing literature for financial topics, we can use one of the available glossaries to identify financial terms in the documents. Then, we can expand the identified terms using one or more of the available financial ontologies and thesauri. In fact, we plan to incorporate many such resources in our framework for a variety of topics, and use them all, irrespectively of the topics that appear in the underlying collection. This will allow us to handle a variety of collections, beyond archives of news articles.

7.2.7 Conclusion

We presented techniques for automatically identifying terms that are useful for building faceted hierarchies over two kinds of collections. For a text-annotated collection, such as Corbis, we built a classifier that can associate, with a high level of accuracy, annotated keywords across a pre-defined set of facets. For a free-text

²³<http://www.taxonomywarehouse.com/>.

collection such as The New York Times archive, we built a set of techniques that develop the idea that external resources, when queried with the appropriate terms, provide useful context that is valuable for locating the facets that appear in a collection of text documents. We demonstrated the usefulness of Wikipedia, WordNet, and Google as external resources. Our experimental results, validated by an extensive study using human subjects, indicate that our techniques generate facets of high quality that can improve the browsing experience for users.

Chapter 8

System Implementation

**Giovanni Maria Sacco, Yannis Tzitzikas,
and Sébastien Ferré**

*“We have too many high sounding words, and too few actions
that correspond with them.”*

Abigail Adams, 1744–1818 (letter to John Adams, 1774)

This chapter discusses system implementation. Section 8.1 provides an analytical discussion of the various implementation choices, and Sect. 8.2 reports experimental results over relational databases. Section 8.3 discusses existing systems and Sect. 8.4 discusses formats that can be used for representing and exchanging taxonomies and taxonomy-based metadata, as well as a brief discussion on protocols. Finally, Sect. 8.5 discusses taxonomies as an Abstract Data Type and shows how taxonomies can be composed.

8.1 Architecture and Implementation Strategies

Real-time response to user navigation is required because a slower execution would severely impair the sense of free exploration that the user of dynamic taxonomy systems experiences. The central operation is the zoom operation, i.e., the computation of the user-defined focus set, the subsequent computation of related concepts and the final reduction of the corpus taxonomy.

Implementations of dynamic taxonomies are not discussed in detail in literature [236, 327]. The majority of current implementations manage a small number of objects described by shallow taxonomies: in this case, any brute force implementation is probably adequate. However, large-scale applications on infobases in the multi-million-object range, described by complex, multilevel taxonomies will gradually become widespread. In this context, real-time zoom operations are indeed a critical issue, and efficient structures and evaluation strategies are required. The architecture described here was introduced by [231–233] and achieves a speedup of more than two orders of magnitude with respect to a standard relational implementation.

In most real-life situations, we expect access by dynamic taxonomies to be a component of a larger existing software infrastructure. In e-commerce applications,

currently the most popular application area for dynamic taxonomies, product selection by dynamic taxonomies has to be integrated with other required components such as shopping cart management and credit-card processing, inventory management, etc. Therefore, dynamic taxonomy systems have to be engineered in such a way as to facilitate integration into existing systems. In fact, the extremely wide application range of dynamic taxonomies suggests that dynamic taxonomy engines should be implemented as a component, or as a Web service, to be invoked by complex applications.

Because of faceted design and of the quick convergence of dynamic taxonomies, the taxonomy itself is expected to be quite compact and taxonomies with one thousand terminal concepts or less to be the norm in practice. Exceptions usually arise when summarization must be supported on large domains, such as authors, titles, etc. As an example, consider a classical music recording database. Although the recordings for specific authors can be directly retrieved by information retrieval queries, representing them explicitly in the taxonomy allows showing a summary of the authors of, for instance, string quartets, and consequently supports a more complete exploration of the infobase, especially for knowledge and wisdom-seeking tasks. These exceptions will be discussed later in the section on virtual concepts.

Finally, even if most current applications are essentially monolingual (English-based, mostly), Internet applications are inherently transnational and the multilingual capabilities of dynamic taxonomies are an important advantage which should be exploited.

8.1.1 Logical Architecture

The goals of the general logical architecture described in the following are flexibility and performance. There are three key points to be considered for a flexible architecture. First, dynamic taxonomies are explicitly concerned with information access only and not with information classification. Objects are assumed to be classified by some third entity, i.e., human beings or automatic/semiautomatic classifiers. It is likely that machine classifiers will need additional structures and/or data, but these will be ignored here, since our only concern is *efficient exploration* through dynamic taxonomies. Consequently, the first separation we make is between information access and information classification.

Second, information access is based on a conceptual (metadata) description of information objects, which is completely independent of the actual object contents [236]. This clean separation between conceptual descriptions and actual contents is not always present in information access literature where, for instance, the conceptual taxonomy is often considered equivalent to a hierarchical thesaurus, with obvious problems and limitations.

Third, the definition of a concept C as a set of objects classified under C indicates that the concept "name" is just a label and there may be different captions, possibly multilingual, for C .

The architecture described here makes a clear separation among several aspects of dynamic taxonomies. First, the identification of an object is separated from its actual content, so that dynamic taxonomies can be used to manage any type of heterogeneous objects. Each object is identified throughout the system through its unique abstract *object identifier* DID, generated by a single system-wide source. Object contents are stored in a *repository*, whose responsibility is to provide storage, extraction and display primitives, based on the object's DID. We will not describe the repository further, because we expect that in most real-life applications it will be already operational and implemented by the host software infrastructure. This provides the second separation: namely, we completely separate the metadata level from the data level and provide interchange between the two levels through object identifiers. Since the medium and type of a specific object are exclusively dealt with by the repository (but can also be represented at the metadata level if required for descriptive purposes), it is clear that the metadata level can effectively abstract from these features: it becomes straightforward to deal with data of any type and format and to interoperate with existing software.

Second, while it is conceptually feasible to represent metadata through a single monolithic structure, such architecture severely impairs the flexibility of the resulting system. The third separation is between the intension and the extension, so that different applications can share the same conceptual schema. Interoperation between the intension and the extension can be provided by identifying each concept by a unique *concept identifier* (CID). Thus the intension will hold the relationships among concepts, while the extension will hold the classification of objects, each identified by its abstract identifier DID, under concepts represented by their abstract identifier CID.

The use of an abstract identifier for concepts also separates concept labels from concept contents (i.e., the set of objects identified by the concept) and from the topological representation of the taxonomy. Concept labels are stored separately in dictionaries, one for each language. Concept labels are to be interpreted in the broadest sense: images, portions of images or icons can be used to identify a concept. The resulting architectural schema is shown in Fig. 8.1.

Since dynamic taxonomies are usually compact, we expect intensions which can be easily stored in RAM. The structures to be used for the intension are basically:

1. a father-to-son structure, FS, which lists for each concept CID the sequence of its sons, ordered by display order. This structure is used to display the taxonomy from the root down;
2. a son-to-father structure, SF, which lists for each concept CID the set of its fathers, or its single father if multiple inheritance is not supported. This structure allows upwards navigation from a concept to the taxonomy root;
3. a DESCENDANTS structure which lists for each concept the set of all its descendants; and
4. an ANCESTORS structure which lists for each concept the set of all its ancestors.

The FS structure is always required, and the DESCENDANTS structure can be dynamically constructed from it, if required. As we will note shortly, upwards naviga-

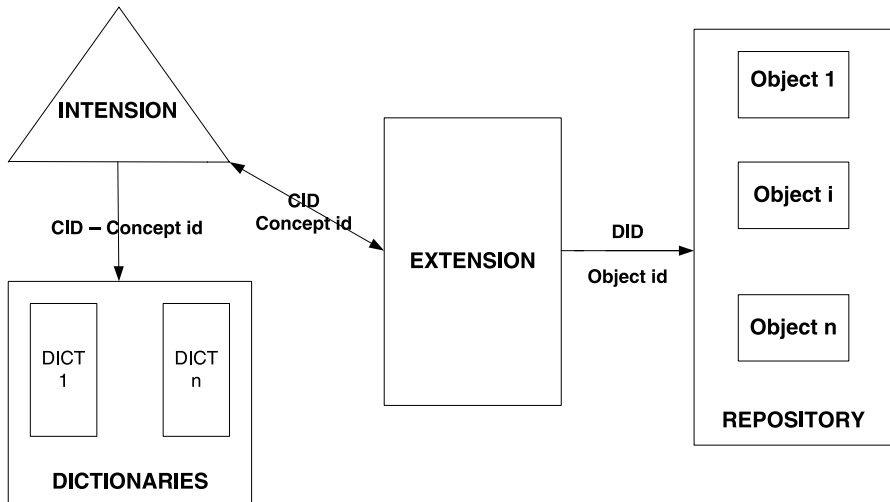


Fig. 8.1 General architecture

tion in the taxonomy is required to implement backwards inheritance.¹ Usually the ANCESTORS structure is used instead of the SF structure. All the stored structures are updated when the intension is modified.

As regards the extension, the principal operation in dynamic taxonomies is the zoom operation. This operation can be expressed in terms of two primitive operations:

1. setting a focus, i.e., computing the set of objects which satisfies a boolean condition on concepts;
2. reducing the current taxonomy, by pruning all the concepts not related to the current focus from it, and, optionally, computing for each concept C the number of objects in the focus which belong to the deep extension of C .

8.1.2 *Computing the Focus*

The computation of the focus, as a boolean condition on concepts, is performed on the deep extensions of these concepts. It is therefore a requirement for performance that deep extensions be explicitly stored. If only the shallow extension is explicitly stored, the deep extension of C must be dynamically computed by taking the union of all the shallow extensions of all the descendants of C . This operation is obviously quite expensive, especially for deep and complex taxonomies and for large information bases. The key to efficient operations is therefore to maintain the deep

¹See Chap. 1.

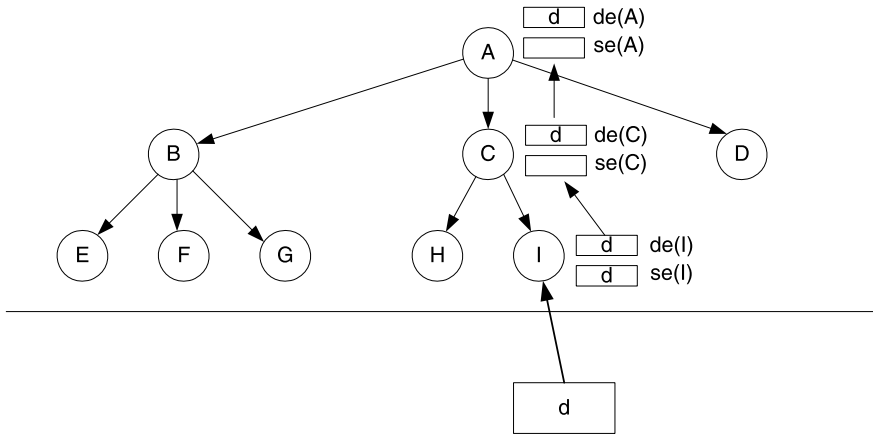


Fig. 8.2 Classification of *d* through backward inheritance

extension of each concept *C* explicitly in the extension. This strategy by itself can improve the speed of operations by orders of magnitude, depending on the taxonomy.

The deep extension can be efficiently constructed during indexing by *backwards inheritance*: if object *d* is classified under concept *C*, then *d* is inserted in the shallow extension of *C* and is also inserted in the deep extension of *C* and of all of *C*'s ancestors (see Fig. 8.2 where “se” denotes shallow extension and “de” deep extension). This method constructs the deep extension bottom-up (from sons to fathers) once, during classification, rather than top-down for each focus evaluation. By construction, the deep extension of the root is equal to the universe, i.e., to the set of all classified objects, and as such might not be explicitly maintained. In order to efficiently implement backwards inheritance, the intensional ANCESTORS structure can be used.

The shallow extension is required only when objects can be classified under non-terminal concepts. In fact, for terminal concepts, the shallow and the deep extension are the same, by construction. In addition, if objects can be classified under terminal concepts only, the shallow extension for non-terminal concepts is trivially empty. For simplicity, we will assume in the following that objects can only be classified at terminal nodes in the taxonomy, so that shallow extensions are not required. The deep extension is stored in the form `DEEPEXTENSION(CID, DID)`, where CID is the concept id, and DID is the object id for each object in the deep extension of concept CID.

One of the important features of dynamic taxonomies is that they can be seamlessly integrated with other search techniques, such as database or information retrieval queries. Dynamic taxonomies can be used in two different ways:

1. to summarize the result of traditional queries, according to the taxonomy;
2. to select a conceptual context through the taxonomy, to which the external search will be restricted.

Integration assumes that the external search mechanism can operate with the same document identifiers used by dynamic taxonomies. This allows to compute the focus set as the intersection of the focus F defined by a combination of concepts from the taxonomy with the result set R computed by the external search mechanism. The resulting focus set may be empty, if the two sets are disjoint, or if R is empty. This accounts for the more general second case above. In the first case the focus set is given by R .

8.1.3 Computing the Reduced Taxonomy

The reduced taxonomy $RT(S)$ can be computed as a function of the focus set S , by pruning from the original taxonomy all the concepts which are not in the related set of S ($RS(S)$). The related set $RS(S)$ can be represented by a set of concept ids which are related to S : this is called the *no-count strategy*. In the *count strategy*, instead, each element of $RS(S)$ contains in addition to the concept id C also the cardinality of $C \cap S$. Such cardinality indicates how many objects in the focus are also classified under C .² Although count strategies are inherently more expensive than no-count strategies, related counts represent an important point in interaction because they indicate whether an additional zoom operation can be beneficial or the infobase is sufficiently reduced to allow the manual inspection of the remaining candidate objects.

The related set $RS(S)$ itself can be computed in two different ways:

1. the focus-driven method
2. the taxonomy-driven method

The *focus-driven* method processes the focus set S and recovers all the concepts used to classify any object in S . In order to compute $RT(S)$, we need an abstract relation which stores, for each object d in the infobase, the concepts used to classify d : $\text{CLASSIFICATION}(\underline{\text{DID}}, \text{CID})$, where DID is the object id of d , and CID is the concept id for each concept used to classify d . The related set $RS(S)$ can then be computed in the no-count strategy through the following SQL query:

```
SELECT DISTINCT T1.CID
FROM RESULT T, CLASSIFICATION T1
WHERE T.DID=T1.DID
```

where $\text{RESULT}(\underline{\text{DID}})$ is the relation which represents the focus set.

The count strategy requires the following more complex SQL query:

```
SELECT T1.CID, CNT(*)
FROM RESULT T, CLASSIFICATION T1
WHERE T.DID=T1.DID
GROUP BY T1.CID
```

²If $|C \cap S| = 0$, then C does not belong to $RS(S)$.

An implementation based on joins is used in Flamenco [99]. According to Yee [327]:

query previews [*i.e.*, *reduced taxonomies*] are generated using the SQL COUNT(*) and GROUP BY operators to count the number of objects which fall into each subcategory.

It is worth mentioning that Flamenco does not store the deep extension of concepts [132] and has to dynamically reconstruct it by taking the union of the shallow extensions of descendant concepts. As we remarked above, this can easily translate into an overhead of more than two orders of magnitude.

The *taxonomy-driven* method [231–233] is based on one of the alternate formulations of the extensional inference rule:

given a concept C and a concept expressed by an arbitrary subset S of the universe, C is related to S , $C \rightleftharpoons S$ iff $objects(C) \cap S \neq \emptyset$

This formulation shows that the related set can be computed by performing, for each concept C in the taxonomy, the intersection between the extension of C and the focus set S . If the intersection is non-empty, then C belongs to the related set $RS(S)$. The taxonomy-driven method only requires the DEEPEXTENSION relation. In the taxonomy-driven method/no-count strategy, the intersection operation between the extension of a concept C and the focus set S can be stopped as soon as a common object id is found. In the count strategy, instead, the entire intersection must be performed.

By remembering that the inclusion constraint in subsumption implies that if $C \notin RS(S)$ and C' is a descendant of C , then $C' \notin RS(S)$, the evaluation of $RS(S)$ can be *optimized* by computing the related set in a top-down way, *i.e.*, from the root towards terminal concepts. Whenever a concept C has a null intersection with the current focus, all of its descendants will have a null intersection as well and they can be discarded without computation. The DESCENDANTS structure in the intension can be used to quickly find all the descendants of C . This optimization is expected to be more effective for smaller focus set cardinalities.

8.1.4 Presentation Strategies

In the previous section, we described how operations on dynamic taxonomies, and especially the extensional inference rule, can be efficiently supported. In this section we discuss presentation strategies for dynamic taxonomies, *i.e.*, how the reduced taxonomic tree is generated. We will assume that the dynamic taxonomy engine is available at a web server, and that the interaction is stateless on the server side.

When the user establishes or modifies the current focus, the existing taxonomy has to be recomputed, in order to prune irrelevant concepts. There are two main strategies here:

1. a *labor-intensive or full-loading strategy*, in which the entire tree is reduced and sent to the client. Any additional exploration not involving additional zooms can be carried out client-side, thus minimizing the number of transmissions;

2. a *lazy or on-demand partial loading strategy*, which only computes the highest levels of the taxonomic tree, i.e., the sons of the root, with other levels being computed on demand. This strategy requires a larger number of smaller transmissions, since the explosion of a node requires getting from the server the filtered set of sons of that node.

Although both strategies can be used by focus-driven and taxonomy-driven methods, the lazy strategy benefits only the taxonomy-driven approach, because the focus-driven approach always computes the entire related set. In the taxonomy-driven approach, instead, each concept is individually tested for membership in the related set, so that if dynamic taxonomies guidelines [236] are met and the taxonomy fanout is limited to 10–20 elements, the response time is usually at least two orders of magnitude faster than the computation of the complete reduced taxonomy, because only the immediate sons are computed each time. In addition, a lazy taxonomy-driven method minimizes the work to be done, because users do not usually explore the entire reduced taxonomy, but only expand a very limited number of concepts in the tree.

With taxonomy-driven methods, the labor-intensive strategy requiring the computation of the entire tree can very well be several orders of magnitude more expensive than the lazy strategy, at least as far as response time is concerned. It is therefore important that the labor-intensive, full-evaluation strategy be optimized, as discussed in the previous section.

8.1.5 Physical Storage Structures for the Extension

Although a relational implementation is viable, special structures can significantly improve the performance of dynamic taxonomies. Here we consider four different structures to represent the deep extension of concepts, i.e., the list of document ids for the objects classified directly or indirectly under a concept:

1. Inverted lists;
2. Bit vectors, similar to the bitmap indices [199] currently supported by several relational DBMSs, such as Oracle and Sybase, for OLAP applications;
3. An opportunistic hybrid compression scheme which uses bit vectors or inverted lists depending on the number of objects to be recorded;
4. Other Bit Vectors compression schemes, such as the Word-Aligned Hybrid (WAH) scheme proposed by Wu et al. [319].

In all of these structures, a pointer array keyed by concept id is used. Each element points to the stored structure for the deep extension of the corresponding concept. We anticipate the following analysis to state that we expect the entire deep extension to be memory-resident, even for very large databases.

In order to simplify the analysis, we assume a faceted taxonomy on j independent facets, in which each object is classified once under each facet. The taxonomy is also assumed to be balanced, i.e., each terminal has the same distance L from the

root. Finally, a constant concept fanout equal to j will be assumed. These assumptions are also used in the experiments described in the following, and represent a fair approximation of common scenarios, such as e-commerce applications. When expedient and for concreteness, a sample infobase of 800,000 objects described by a taxonomy with a constant fanout of 10 and 1,000 terminals is used.

In considering alternate storage schemes, the primary concern is execution speed, because of real-time response requirements. For this reason, data compression methods such as LZ77 ([338], used in gzip) are not considered, as they considerably slow down logical operations.

Inverted lists are widely used in database and information retrieval systems. The inverted list representation for the DEEPEXTENSION relation stores, for each concept C , an ordered vector of object ids, such that an object id DID is in the list if DID is in the deep extension of C . Since object ids are kept sorted in the list, boolean operations are efficiently implemented by merging, with a linear time complexity. The storage required is roughly half of the storage required by the raw data in DEEPEXTENSION, because the concept id CID for a set of entries $\{CID, DID_i\}$ is stored only once. Under our assumptions, the total space requirements for the deep extension stored as inverted lists is

$$S = w(N + jLN)$$

where N is the total number of objects and w is the size in bytes of an object id (usually $w = 4$). In fact, N entries at each level of each facet are entered (jLN entries). In addition, also the root has N entries. For the sample infobase, this results in approximately 96 Mbytes, easily stored in RAM with current technology.

In the bit vector implementation, the extension is stored as a bit matrix B of K rows by N columns, where K is the total number of concepts in the taxonomy and N is the total number of objects in the corpus. $B[i, j]$ is set iff the object d such that its abstract identifier DID is j is in the deep extension of concept C whose abstract identifier CID is i . The implementation of all the set operations required to set a focus, and of the intersection between two concepts required to implement the extensional inference rule can be performed by logical operations on bit vectors in linear time. Storage requirements for the deep extension of an infobase with N objects classified under K concepts is

$$S = KN/8$$

In the example, bit vectors require 108 Mbytes.

We compare the space requirements of the two methods, in order to derive the condition for inverted list storage to be smaller than bit vector storage:

$$wN(1 + jL) \leq KN/8 \quad \text{or} \quad K \leq 8w(1 + jL)$$

This condition does not depend on the number of objects stored, but rather on the total number of concepts and on the number of facets and of levels in the taxonomy. For the sample infobase, inverted lists require less space when $K > 992$.

Table 8.1 Space requirements for Inverted List, Bit Vector and Compressed representations. IL: Inverted List; BV: bit vector. Sizes are in 10^6 bytes

Level	Concepts on this level	IL total size	BV total size	Compression type	Compressed total size
0	1	3.2	0.1	BV	0.1
1	10	32	1	BV	1
2	100	32	10	BV	10
3	1000	32	100	IL	32
Total $L = 3$	1111	99.2	111.1		43.1
4	10000	32	1000	IL	32
Total $L = 4$	11111	131.2	1111.1		75.1

The motivation of strategies 3 and 4 is that bit vectors for concepts become denser for concepts higher in the taxonomy. By definition, the bit vector of the root is all set, while the bit vector for a terminal is expected to have a very small number of bits set. Thus, from the one side, there are opportunities for compressing the higher levels of the tree (strategy 4). On the other hand, the deep extension of the lowest levels of the taxonomy, which is responsible for most of the space requirements, tends to be very sparse, and could be significantly compressed by storing document ids explicitly in the form of an inverted list, rather than as a bit vector (strategy 3).

Assuming a taxonomy on j independent facets with a constant concept fanout equal to j and L levels, the average number of entries for a concept at level i is

$$N/j^{(i-1)}, \quad 1 \leq i \leq L$$

The representation through an inverted list is then less expensive than the corresponding bit vector when

$$wN/j^{(i-1)} < N/8 \quad \text{or} \quad j^{(i-1)} > 8w$$

Consequently, the compression strategy uses uncompressed bitmaps at the highest level of the tree, and inverted lists at the lowest. Table 8.1 reports storage costs for strategies 1, 2 and 3 on a sample infobase with 800,000 objects described by a taxonomy with a constant fanout of 10. Even though the number of terminal concepts found in most practical situations rarely exceeds 1000, we have considered a taxonomy with 1,000 and a taxonomy with 10,000 terminals. From the total space requirements, we find that the compressed total size is 43% of the Inverted List storage and 39% of the Bit Vector storage, in the first case. It is 57% of the Inverted List storage and 7% of the Bit Vector storage, in the second case. For increasing number of levels, the size of the Compressed representation tends to the size of the Inverted List representation, since only the very highest levels in the tree will be managed by Bit Vectors, and these levels account for less and less space for increasing depth of the taxonomy. However, 4 to 5 levels are rarely exceeded in practice.

Boolean operations on the Compressed representation require additional care when the operands are of different types. The AND of the deep extension of two concepts C and C' , where $objects(C)$ is represented by a Bit Vector and $objects(C')$ by an Inverted List is solved by starting from the Inverted List and, for each $d \in objects(C')$ testing if d belongs to $objects(C)$. If the test is successful, C is inserted in the result, otherwise it isn't. The result is in the Inverted List format, which is desirable since the result size will be no larger than $|objects(C')|$ for which an Inverted List representation was beneficial.

The OR of the deep extension of two concepts C and C' , where $objects(C)$ is represented by a Bit Vector and $objects(C')$ by an Inverted List is solved by copying $objects(C)$ to the result, and, for each $d \in objects(C')$ setting the bit corresponding to d in the result. The result is in the Bit Vector format, which is desirable since the result size will be no smaller than $|objects(C)|$ for which a Bit Vector representation was beneficial.

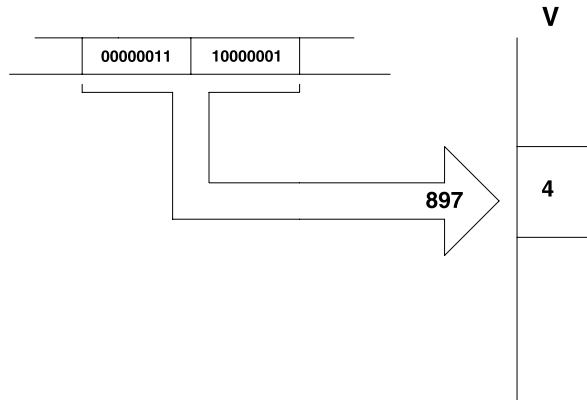
Other Bit Vector compression methods cannot be easily combined with other methods and execute boolean operations slower than Inverted Lists and uncompressed Bit Vectors. The only reason to consider them is if we expect significant storage savings. However, the results reported in [319] for WAH (Word-Aligned Hybrid compression) indicate a compression ratio of less than 50% with respect to Inverted Lists, which is in the same range as the compression we proposed. WAH is not the most efficient compression method, but other methods such as BBC [24] are known to execute boolean operations significantly slower [319].

As mentioned before, the taxonomy-driven method only requires the deep extension of concepts, because it computes the related set $RS(S)$, by intersecting the deep extension of each concept C with the focus set S .

In the no-count strategy, the intersection operation can be stopped as soon as a non-empty intersection is produced. The count strategy requires instead that the number of objects in the intersection result be counted. This is an expensive operation. A naïve counting algorithm works byte-by-byte on the result of the computed intersection. For each byte, shift-mask-test can be performed for each bit, increasing the counter if the bit is set. This algorithm is quite inefficient especially for large sets. The Count strategy on Bit Vectors can be optimized by performing counting in parallel with the evaluation of set operations, in the following way (see Fig. 8.3). We precompute a byte vector V of size 2^k , which for each $V[i]$ holds the number of bits set in the number i . In practice, $k = 16$ will be used and V will have 65,536 elements, one for each different bit configuration of a word. The intersection between two binary vectors is likely to be computed by logical operations on chunks of 4 bytes: then, as soon as a resulting 4-byte chunk is computed, each of the 2-byte chunks can be used as an index to V and the corresponding value of V used to increase the counter.

As an additional optimization, it should be noted that the result of the intersection needs not be stored, since only its cardinality is required in the related set.

Fig. 8.3 Using a precomputed vector to count the number of objects in a bit vector



8.1.5.1 Analysis of Taxonomy and Focus-Driven Evaluation

In comparing taxonomy-driven with focus-driven methods, we first note that taxonomy-driven methods require roughly half the storage required by focus-driven methods. In fact, taxonomy-driven methods only require the DEEPEXTENSION relation both for the computation of the focus set and for the computation of the related set. Focus-driven methods still require the DEEPEXTENSION relation for the computation of the focus set, and the additional CLASSIFICATION relation for the computation of the related set. Taxonomy-driven evaluation is therefore the choice when main memory storage is at a premium, because using secondary storage will generally produce a significant performance degradation with respect to main memory solutions.

Intuitively, we expect focus-driven methods to be more efficient for smaller focus set cardinalities. The behavior of taxonomy-driven evaluation depends on the count/no-count strategy used. For no-count strategies, taxonomy-driven evaluation becomes more efficient as the focus set cardinality grows larger, because the probability of early detection of non-empty intersections increases. In count strategies, unoptimized evaluation is constant, whereas optimized evaluation becomes more efficient for smaller focus sets, because subtree pruning becomes more effective. Finally, lazy taxonomy-driven strategies, though constant in cost, are faster than labor-intensive strategies by one or more orders of magnitude.

In order to initially compare the two methods, we compare the cost of evaluation of the related set for a count unoptimized taxonomy-driven evaluation (S1) in which the deep extension is stored as a bitmap vector, with the cost of a count focus-driven strategy (S2) in which the CLASSIFICATION relation is represented as an inverted list structure. We compute the cost as a function of the number of accesses to 4-byte words, and keep the same assumptions made before.

The cost of S1 is given by $cost(S1) = K \cdot 2(N/32)$. In fact, the unoptimized strategy computes, for each of the K concepts, the intersection between the focus

and the deep extension of the concept. Each bit vector is $N/32$ words long, and the intersection requires the access to both vectors.

Strategy S2 joins the focus set with the CLASSIFICATION relation in order to extract a number of concepts NC equal to $NC = |S|Lj$, where L is the depth of the taxonomy and j the number of facets. These concepts are then sorted with an $n \log n$ cost, and finally sequentially scanned to compute the result. Let $N' = N/\delta = |S|$, $\delta \geq 1$, the cost of S2 is given by $cost(S2) = 2N'Lj + 2N'Lj \log_2(N'Lj) + NLj$. The cost of S2 is smaller than the cost of S1 if

$$\delta > \frac{16Lj(3 + 2 \log_2(N'Lj))}{K}$$

In the case of a lazy, on-demand evaluation, only j out of K concepts are going to be tested, so that the cost of S2 is smaller than the cost of S1 lazy if

$$\delta > 16L(3 + 2 \log_2(N'Lj))$$

Figure 8.4 reports the estimated break-even point for focus-driven strategy S2 with respect to taxonomy-driven on-demand lazy strategy S1, for the sample infobase. S2 is better for focus set sizes smaller than the break-even point. Figure 8.5 reports the estimated ratio between the focus-driven strategy S2 with respect to taxonomy-driven on-demand lazy strategy S1, for the sample infobase. The overall ratio (i.e., considering all the concepts except the root as a possible focus) and the ra-

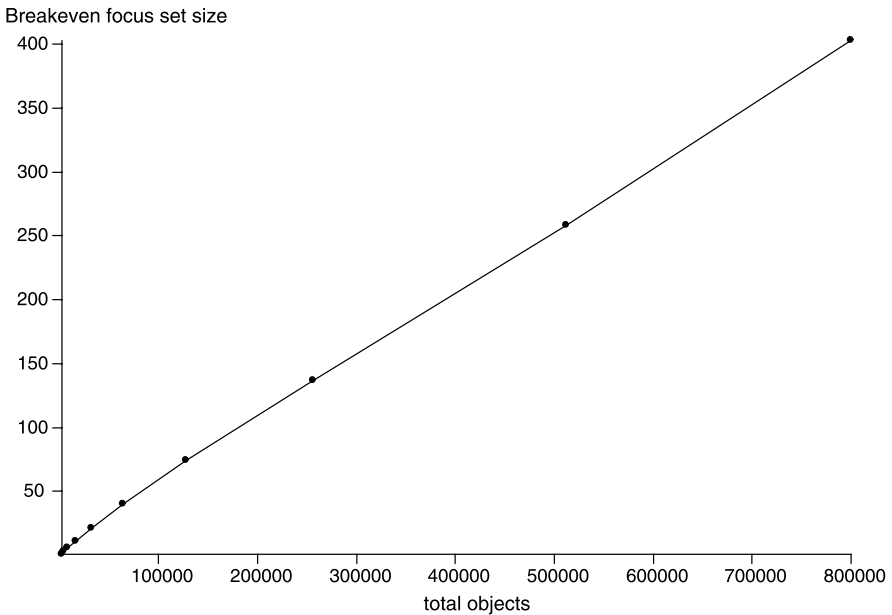


Fig. 8.4 Break-even result set cardinality for taxonomy and focus-driven evaluations

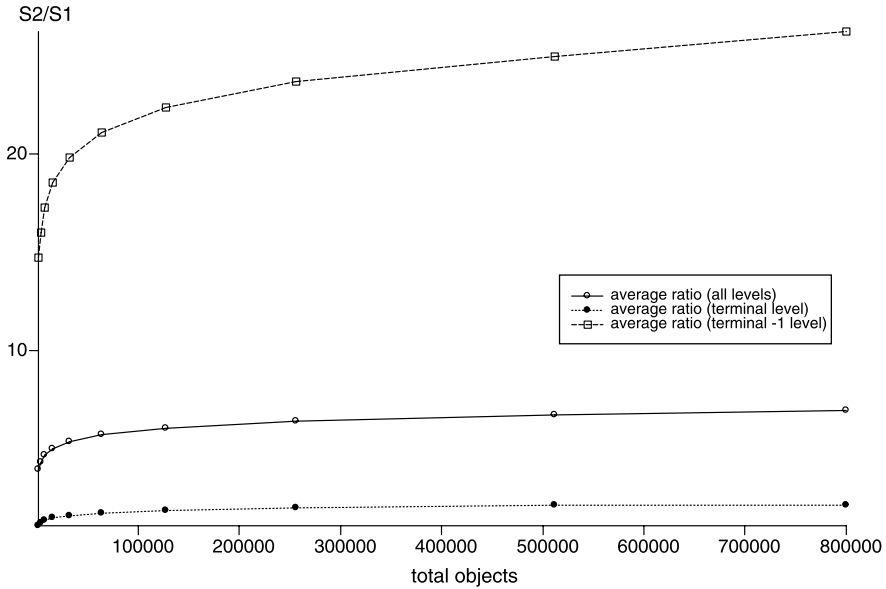


Fig. 8.5 Estimated cost ratios of focus-driven over taxonomy-driven lazy evaluation

tio at the terminal, and immediately higher, levels are reported. The lazy taxonomy-driven strategy is shown to be always better, for the sample infobase, than the focus-driven strategy, and, as expected, larger improvements occur for the higher, denser levels of the conceptual tree.

8.1.6 Experimental Data

In order to verify the strategies described above, we conducted experiments on a synthetically generated sample. The sample simulates a faceted classification on 10 independent facets, each organized on three levels with a fixed fanout of 10. The total number of terminal concepts is thus 1000, while the total number of concepts in the taxonomy is 1111, including the root of the taxonomy which contains all the documents, is explicitly stored but not used in the following experiments. Each document is classified exactly once under each facet, at the terminal level. Although this taxonomy is quite compact, it is a good representative of taxonomies designed following the guidelines reported in [236].³ The experiments were conducted on corpora ranging from 2,000 to 800,000 documents, with the corpus size increasing by a factor of 2 in each experiment. All the possible reduced taxonomies for single-concept foci were computed. This means that we concentrate

³See Sect. 7.1.

on the very first and heaviest phase (zoom) of the iterative thinning of dynamic taxonomies.

Our measures concentrate on the production of reduced taxonomies, since this is the most critical operation in the model. We report our results in reduced taxonomies per second (rtps), based on a Dell Dimension 8250, Intel Pentium 4 2.8 GHz, 512 MB RAM, Microsoft Windows 2000 Professional, an average machine with current technology. The reader is cautioned that our measures do not include the overhead due to HTTP processing, context switches, etc. We only measure low-level implementation costs, because other overheads depend on the Internet server, operating system, etc. and will be the same for all methods. The first experiments compare three different implementations of bit vectors with counters (count by shift-and-mask, by 1 byte table access and by 2 byte table access) with an in-memory implementation of inverted lists. This latter strategy is a pointer array, with each pointer addressing an ordered integer vector, and is significantly faster than secondary memory implementations based on hashing or B-trees.

The first experiment was conducted on the full-evaluation, labor-intensive strategy, which fully evaluates the reduced taxonomic tree. The results are reported in Fig. 8.6 and they show that 2 byte table count bit vectors clearly outperform the other strategies. It is worth noting that this strategy has a throughput of 2.67 reduced trees per second with 800,000 documents. Required main memory for the extension ranged from 273 KB to 106 MB, with bit vector methods requiring about 12% more memory than inverted lists. One byte table count bit vectors were included in the comparison because it was contended that they might be faster than 2 byte table

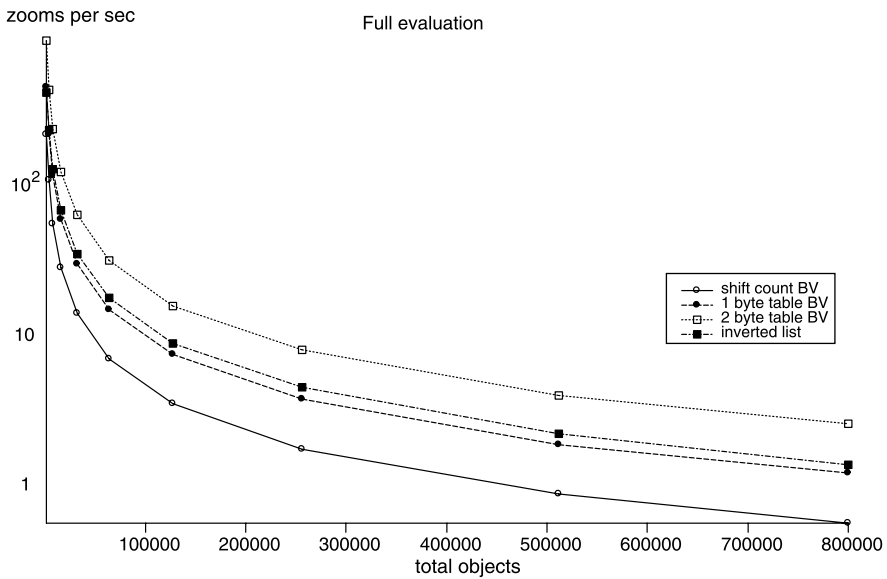


Fig. 8.6 Full evaluation for selected strategies (logarithmic y axis)

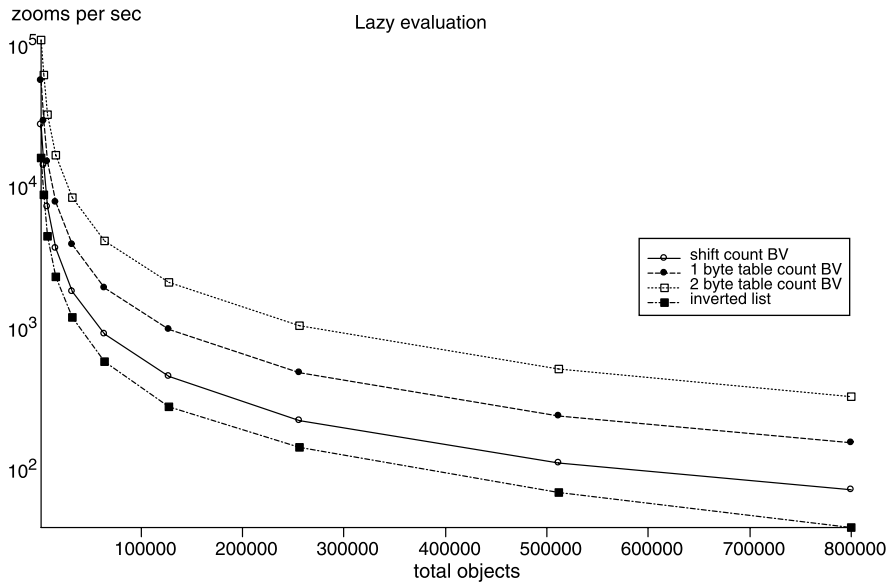


Fig. 8.7 Lazy on-demand evaluation for selected strategies (logarithmic y axis)

count bit vectors, due to a better cache locality. Our experiments show that this is not the case in our environment, and we believe that this holds in general.

The second experiment, reported in Fig. 8.7, compares the same methods on the lazy strategy, in which only the reduced top level of the taxonomy is computed. Again, the 2 byte table count bit vector strategy outperforms the other ones, and it achieves a throughput of 327 rtps with 800,000 documents. Note that the inverted list strategy is now outperformed by all the bit vector strategies. This is because the higher levels of the taxonomy are denser and consequently the corresponding inverted lists longer and more expensive to process. In fact, the speed ratio between 2 byte bit vectors and inverted lists varies from 6.75 (2,000 documents) to 8.35 (800,000 documents), which indicates performance degradation for inverted lists, as the lists grow longer.

The third experiment compares 2 byte table count bit vectors with the no-count bit vector strategy which simply tests the intersection of two vectors for emptiness. These two strategies are applied both to full (Fig. 8.8) and lazy (Fig. 8.9) evaluation. While it is not surprising that the no-count strategy outperforms every counting strategy, it is interesting to note that the reduced trees per second offered by the no-count strategy for 800,000 documents range from 147 (full evaluation) to 23,244 (lazy evaluation). The no-count strategies works better for denser levels, because non-empty intersections tend to be detected earlier.

The fourth experiment was conducted to test the improvements deriving from top down evaluation with empty concept detection. Both 2 byte count bit vectors and inverted lists were considered. The results, reported in Fig. 8.10, show that there is

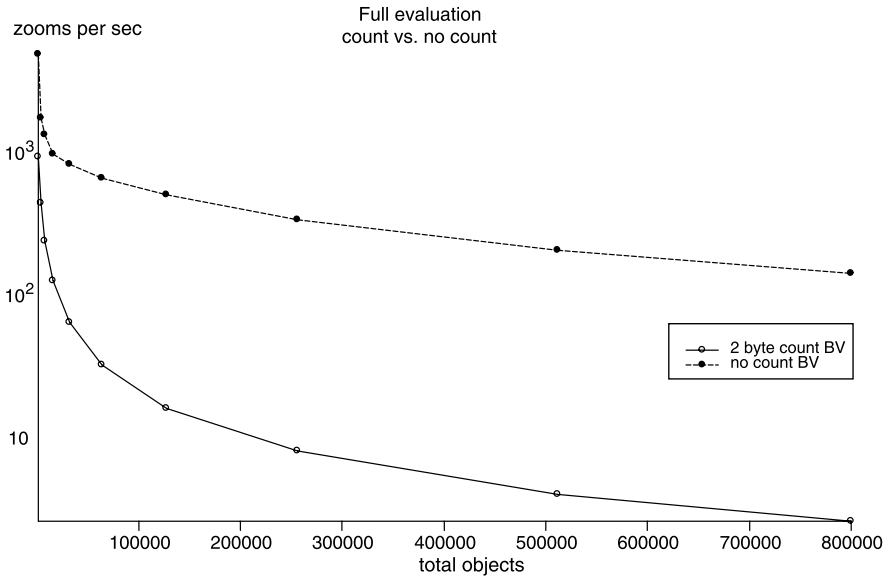


Fig. 8.8 Count and no count strategies for full evaluation (logarithmic y axis)

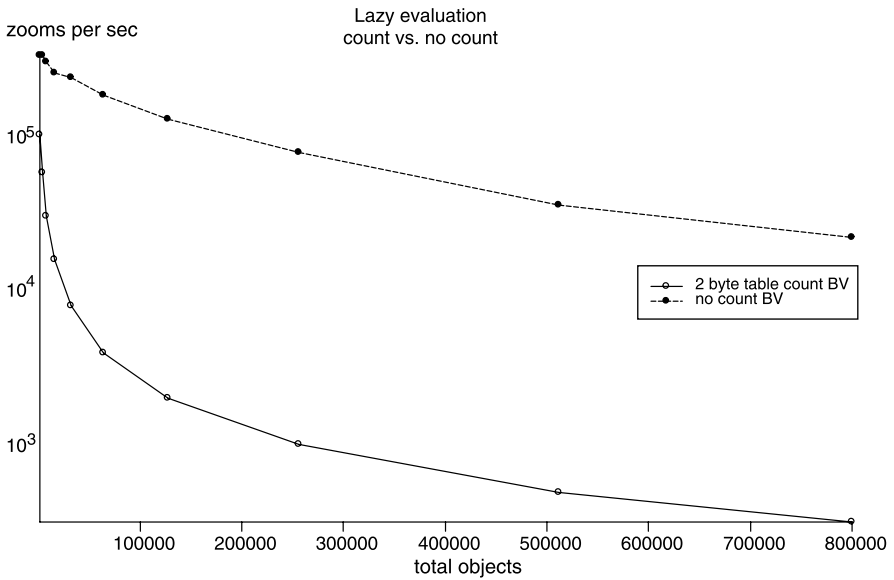


Fig. 8.9 Count and no count strategies for lazy on-demand evaluation (logarithmic y axis)

indeed a significant improvement for light loads (267% at 2,000 documents), which becomes negligible for higher loads. Inverted lists are outperformed also in this case.

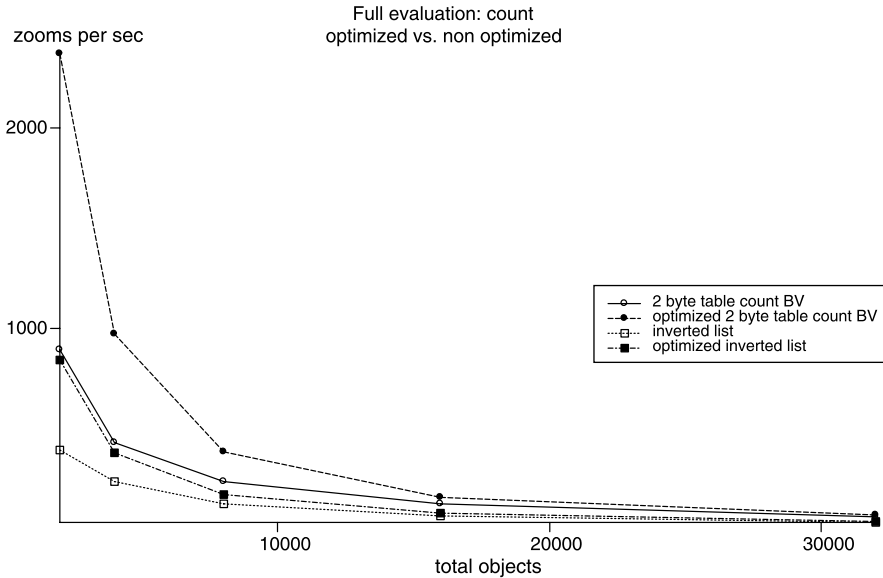


Fig. 8.10 Optimized and non optimized full evaluation

Finally, we have conducted full evaluation experiments to compare our bitmapped (2 byte table counters, conservatively non optimized) architecture to a plain relational database implementation. The experiments were run using MySQL v. 4.0.20ant, on the same synthetic information base used for the other experiments. We created the relation DEEPEXTENSION as a memory-resident table through

```
Create Table DEEPEXTENSION (
  did INTEGER, cid INTEGER
  Primary Key (did, cid)
  KEY(CID) ) TYPE=MEMORY
```

As before, we computed reduced taxonomies for all the possible single-concept foci. The reduced taxonomy is computed by the following query:

```
SELECT T1.cid, COUNT(*)
FROM DEEPEXTENSION T, DEEPEXTENSION T1
WHERE T.did=T1.did AND T.cid=focus-concept_cid
GROUP BY T1.cid
```

Memory requirements for the DEEPEXTENSION relation were over 5.50 times larger than our bitmapped implementation. This was to be expected because

- concept identifiers are not factored out of the deep extension table, so that the size of the relational table is twice as large as the bitmapped table,
- two indices (one clustered and one unclustered) are defined in the relational implementation and

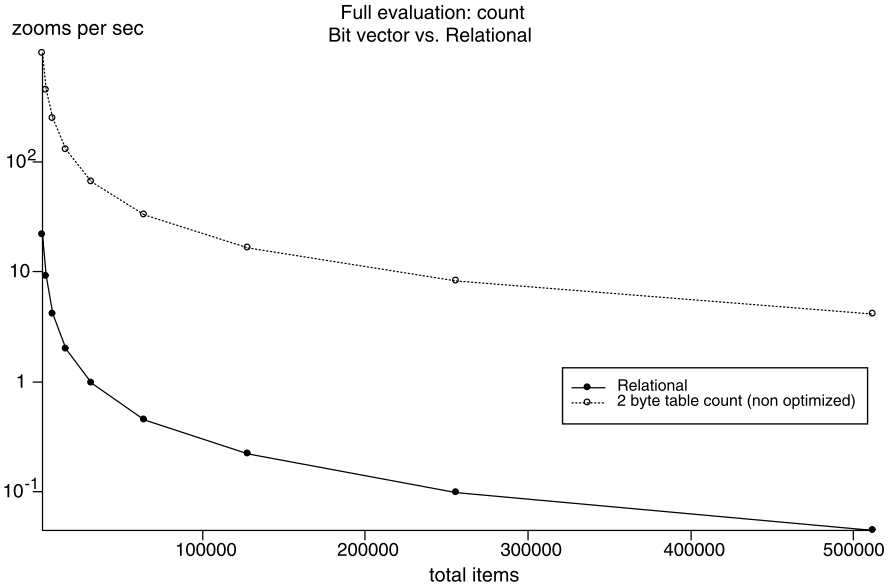


Fig. 8.11 2 byte table count nonoptimized bit vector and relational evaluation

- the dynamic relation and its indices, stored as B+ trees, have an average utilization factor of 0.69.

Obviously, such a large storage overhead significantly reduces the size of information bases which can be held in main memory.

The ratio between rtps for bitmapped and relational evaluation goes from 42.43 for a 2,000 object database to 91.73 for a 512,000 object database (rtps for the two methods are reported in Fig. 8.11). This indicates that the performance of the relational implementation degrades as the database size grows. The throughput for a bitmapped evaluation on 512,000 objects is equivalent to the throughput of a relational implementation on 8,000 objects. Although the experiments depend on the specific RDBMS used, so that the actual throughput for other systems might be different, we are confident that bitmapped evaluation will outperform relational implementations.⁴

It is worth mentioning that the Flamenco implementation [99] is considerably less efficient than the one described above since it does not store the deep extension of concepts [132] and has to dynamically reconstruct it by taking the union of the shallow extensions of descendant concepts. As we remarked above, this can easily translate into an overhead of more than two orders of magnitude.

Access through dynamic taxonomies is beneficial even when the number of documents to be accessed is very small. However, their superior, guided convergence to

⁴There are other factors, beyond performance, which can justify the use of a RDBMS. See Sect. 8.2.

small result sets makes them a clear winner over traditional methods for large and very large information bases. In this context, dynamic taxonomy systems must offer real-time interaction to a potentially large number of users.

We have shown that real-time dynamic taxonomies can be supported by common hardware configurations if specialized software architectures are used. Bit vectors were shown to be an efficient method, although they require slightly more memory space than inverted lists.

The experiments indicate that different strategies can be used, in centralized architectures, as the corpus size grows. Optimized full evaluation can be used for smaller corpora, lazy evaluation for medium to large corpora and no-count lazy evaluation for very large corpora. Memory requirements are rather moderate and a 1 GB RAM configuration can store the required dynamic taxonomy structures for several million documents.

8.1.7 Further Performance Enhancements

Especially in the case of a stateless interaction, substantial computational savings can be obtained by caching focus sets and related sets [231, 233]. We expect that all the different foci are not equiprobable. In particular, we believe that, for each facet, only a subset of its concepts will be used for focusing, the remaining ones being used for summarization. As an example, consider an e-commerce application with a Price facet divided into Budget, Medium, and High. Although possible, it is highly unlikely that the first user focus will be Price > High. If Price is no objection, the first focus will probably be on some other feature, say high resolution. So we believe that if Price is used to set the first focus, only Price > Budget is likely to be used. The same holds for most other available features. As in the end-game for e-commerce we discuss in Chap. 9, we believe that there is a ranking of concepts for each facet, from most desirable to less desirable ones (all other features being equal), and that only the most desirable concepts are likely to be used for focusing.

If this is true, then caching related sets for high-probability foci would require limited storage resources (because relatively few of all the possible foci occur) and at the same time achieve a considerable speedup (because high-probability foci need not be computed). The caching of related sets seems more beneficial than the caching of foci especially in full-evaluation strategies, because the reduced taxonomy can be immediately transmitted to the client without any additional computation.

Although high-probability foci can be probably be determined *a priori* in some application domains such as e-commerce, it may be quite difficult and time-dependent in other domains. For these reasons, we believe that intelligent cache replacement policies that account both for popularity and time, such as the well-known LRU-K replacement policy [198], are especially suitable for this context.

The hit-ratio of such caching strategies depends on the application and, as in virtual memory systems and database buffer managers, on the size of the cache. However, by similarity with VM systems, hit ratios over 80–90% can be expected, which would translate into a 5–10× speedup with respect to non-caching implementations.

Other improvements can be expected by parallel architectures. Viable strategies include replication, vertical partitioning (different sets of documents are allocated to different machines) and horizontal partitioning (different concepts are allocated to different machines, i.e., the taxonomy is partitioned among different machines). These strategies can be combined. Because our implementation requires small to moderate amounts of main memory, we expect that most practical applications can be parallelized by full replication on N machines. In this way, communication overhead is minimum because the only requirement is that requests be balanced among the available machines: we expect a speedup factor close to N . Only very large, multi-million document applications will require some kind of partitioning.

8.1.7.1 Virtual Concepts

Simple virtual concepts and *derived virtual concepts* were introduced in Sect. 5.3 as a way to “virtualize” parts of the taxonomy and materialize such parts, when appropriate, from additional, external structures.

A simple virtual concept \bar{V} (e.g., Price) is fully characterized by four abstract operations:

1. Given \bar{V} , find all its sons. In our example, this means finding all the distinct values of Price in the infobase;
2. Given \bar{V} , find its deep extension: e.g., find all the objects for which Price is defined;
3. Given the son s of \bar{V} , find its deep extension. In the example, find all the objects which have the specified value s of Price; and
4. Given an object d , find all the descendants of \bar{V} under which d is classified. In the example, find all the Prices for a specific object.

A way in which these abstract operations can be implemented is to keep a relation $C[\bar{V}](\text{Value}, \text{DID})$ for each virtual concept \bar{V} , and a secondary index on $(\text{DID}, \text{Value})$. In this way, the four operations can be implemented by the following SQL queries:

1. SELECT DISTINCT Value FROM $C[\bar{V}]$
2. SELECT DISTINCT DID FROM $C[\bar{V}]$
3. SELECT DISTINCT DID FROM $C[\bar{V}]$ WHERE Value = s
4. SELECT DISTINCT Value FROM $C[\bar{V}]$ WHERE DID = d

Counting is easily added. Note that $C[\bar{V}]$ need not be explicitly stored but can be synthesized on request. This implies that the Value considered here can be derived from external data by, for instance, statistical operations such as means, totals, etc. On the other hand, if $C[\bar{V}]$ is actually stored, multiple $C[\bar{V}]$, one for each virtual

concept \bar{V} , can be stored in the same relation. When the reduced taxonomy for a focus F is constructed, the operations are computed on $C[\bar{V}] \bowtie_{\text{DID}} F$.

A derived virtual concept $\delta(\bar{V})$ can be derived from a virtual concept \bar{V} by specifying additional restrictions on the (base) relation $C[\bar{V}]$ in the WHERE clause of each SQL query.

For example, consider a virtual concept “Price”. Rather than exploding “Price” into its actual values, we might want to group these values in a small number of ranges. Assume that one such range is “Budget”, defined as $\text{Price} < 50$. Then the four operations required to characterize the derived virtual concept “Budget” are the same as for Price, with the addition of the clause $\text{WHERE Value} < 50$.

Derived virtual concepts can be derived from other derived virtual concepts, in order to define an arbitrary hierarchy: in this case the additional restrictions are composed in AND. Derived concepts are dynamic in nature, and the end-user could be enabled to dynamically specify custom groupings through parametric restrictions.

8.1.7.2 Time-Varying Concepts

In many applications, and notably online auctions, part of the classification for an object depends on time [231–233]. For instance, an object can be classified for time to auction end, usually according to predefined intervals (one hour or less, one day or less, etc.). Such time-varying concepts can be represented by virtual concepts in the following way. A time instant t is represented as an abstract timestamp, containing the number of clock ticks from a fixed time origin; the clock resolution depends on the application. The difference between two timestamps t and t' defines the time interval between the two times. We can define a virtual concept \bar{V} whose values are the set of timestamps of all documents in the extension of \bar{V} . Let T be the timestamp of the current time, and the sons of \bar{V} be represented as time intervals with respect to the current timestamp T . The abstract operations for \bar{V} can be defined as:

1. Given the virtual concept V , retrieve all its sons: $\text{SELECT DISTINCT } T\text{-value FROM } C[\bar{V}]$
2. Given the virtual concept V , retrieve its deep extension: $\text{SELECT DISTINCT DID FROM } C[\bar{V}]$
3. Given the son s of the virtual concept V , retrieve its extension: $\text{SELECT DISTINCT DID FROM } C[\bar{V}] \text{ WHERE value} = T - s$
4. Given an object d , find all the descendants of V under which a is classified: $\text{SELECT DISTINCT } T\text{-value FROM } C[\bar{V}] \text{ WHERE DID} = d$

With virtual concepts, the classification of objects among intervals is dynamically computed when the deep extension of an interval is required. An alternate way is to split the values of a time-varying concept into N intervals (from more recent to older ones), which are represented as real concepts. In this way, the deep extension of each interval is pre-computed, but the classification of each document has to be periodically recomputed. In order to minimize the number of objects which have to be checked for reclassification, we keep, for each interval I :

1. the list $L(I)$ of pairs (DID, timestamp) in the interval ordered by decreasing timestamps (i.e., newer to older);
2. in central memory, an interval representative $IR(I)$, which is the DID in the interval with the oldest timestamp together with its timestamp; and
3. a classification criterion (e.g., T -value less than 1 week and no smaller than 1 day)

When objects need to be reclassified (i.e., after P ticks, where P is selected by the system administrator), we will check, for each interval I , its interval representative $IR(I)$. If $IR(I)$ is null or meets the classification criterion for I , no documents in I need to be reclassified. Otherwise, we will reclassify $IR(I)$, select a new representative $IR(I)$ by picking the last (oldest) element in $L(I)$ and repeat the test. The list $L(I)$ can be implemented as a queue, with new objects added from one end, and old objects extracted, as interval representatives, from the other end.

It is easy to prove that we only consider, for each interval, only one object in addition to those which have actually to be reclassified. This latter strategy is expected to be much faster when access by time-varying concepts is frequent and the reclassification time interval P is reasonably large (i.e., a few minutes).

8.2 Implementation over a Relational Database Management System

Since relational database technology is the de facto standard for business applications here we examine the case where all (meta)data are stored in a relational database. We examine this case because sometimes the database is already in place and its contents may change frequently. For such cases, it is worth investigating the case where exploratory search is implemented on top of SQL and no special index (like those described in Sect. 8.1) is created/maintained. This is the approach followed by systems like Flamenco [327] and i411 (also see Sect. 8.3).

Suppose that we have a relational database and we want to offer exploration services for its contents. One approach is to define a view (by exploiting the declarative query language of the DBMS) containing the attributes that should be considered as facets. This view may comprise attributes coming from different relations and its definition may include joins and other transformations. Each attribute is considered as a facet, and the set of distinct values of these attributes that appear in the tuples of the view are considered as the terms of that facet. As it has been stressed already, guided exploration can be combined with other access methods (e.g. predefined query forms or plain SQL query answering) to summarize the results of these methods. This means that it is not necessary that the faceted view includes all attributes that characterize an object, or all the attributes that are being exploited by the rest access methods; it can contain only those that are appropriate for exploration. However, attributes like “price”, “weight”, “dates”, “locations” often have a big number of distinct values. In such cases an additional step aiming at abstracting/grouping the set of values is appropriate. Sometimes there may exist such a

hierarchy that is already represented in a separate relational table (recall the design strategies for OLAP data warehouses, such as star schemata). If a hierarchy is not available it can be constructed manually or derived automatically. For the latter case automatic methods for defining hierarchies, e.g. for defining intervals for prices, can be adopted. For instance, [61] describes methods for creating multi-level taxonomies for attribute values on the fly. However, there is a trade-off between the degree of automation and the quality of the produced hierarchy.

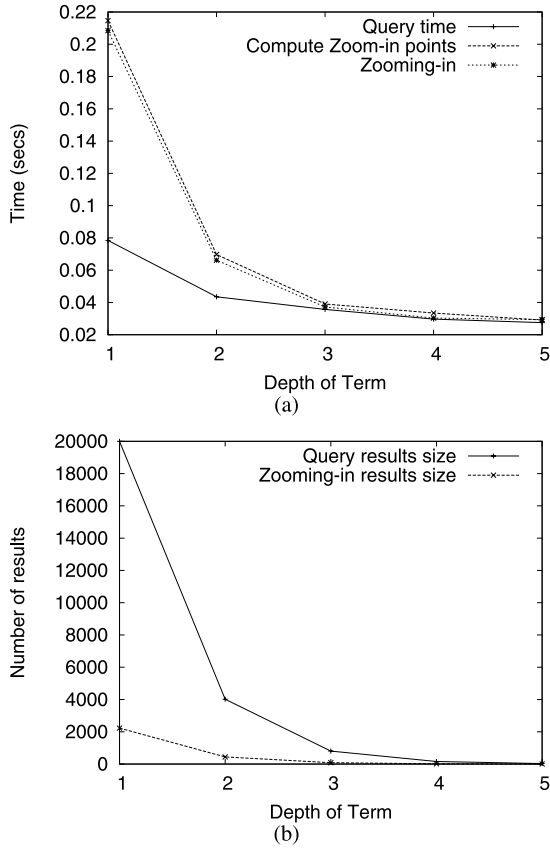
Regarding implementation let's hereafter consider the case where all (meta)data are stored in a (R)DBMS and taxonomies (i.e. their subsumption relations) are represented in the database in tables of the form (son-to-father) only. If only the reflexive and transitive reduction of \leq and the interpretation I (or equivalently the description function D_I) is stored, then the computation of the zoom points can be done with one SQL query only if we a priori know the depth of the taxonomies involved, or if we adopt recursive SQL. Otherwise, more than one queries have to be issued. If the model interpretation \bar{I} (or equivalently the complete description function $D_{\bar{I}}$) is represented explicitly in the database, then one SQL query is enough, but to maintain \bar{I} (and $D_{\bar{I}}$) after updates in the database requires knowing the depth of the taxonomies or using more than one queries.

Table 8.2 shows the schema of a bibliographic database storing information about authors and papers where the latter are classified according to a subject hierarchy. To measure efficiency we created a synthetic data set over this schema. The relation `subjectHierarchy` forms a balanced and complete tree with degree 5 and depth 5. Each paper (through the relation `paperSubjects`) is associated with one randomly selected subject term (that is a leaf), and (through the relation `paperAuthors`) with 1 to 4 randomly selected authors. All fields of the tables

Table 8.2 Schema of the synthetic bibliographic database

Relation	Attribute	Number of tuples
paper	<u>pid</u> title year venue	10^5
author	<u>pid</u> <u>authorName</u>	4×10^4
paperAuthors	<u>pid</u> <u>authorId</u>	2.2×10^5
subjectHierarchy	<u>stId</u> name parentID	3906
paperSubjects	<u>stId</u> <u>pid</u>	10^5

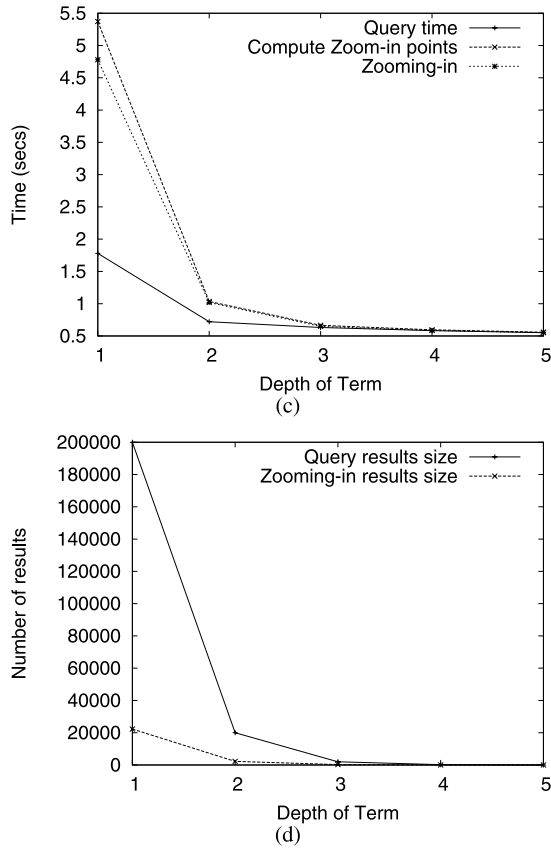
Fig. 8.12 Experimental results on synthetic databases



are indexed with B-trees and the size of the database is 30.1 MB (the indexes occupy 17.2 MB). The experiments were performed over PostgreSQL 8.3 (with shared buffers parameter set to 1 GB) on a Pentium IV machine with 3 GHz CPU and 1 GB RAM. The database is redundancy free, in the sense that only the transitive and reflexive reduction of the subject hierarchy is stored, and only the direct (not the complete) descriptions of the papers are stored. Recall that this storage policy is advantageous in terms of storage space and maintenance, but requires hierarchy traversals (performed with more than one SQL queries as discussed earlier).

Figure 8.12(a) shows: (t_a) the time for computing the answer of a query comprising one subject term from various term depths, (t_b) the time to compute the immediate zoom-in points with respect to the venue attribute (including count information), (t_c) the time to compute the content of the new focus (we have selected one zoom-in point from venue facet). Note that the cost t_a is included in both t_b and t_c , since we re-compute the results. The reported times are the average of 20 different runs of 5 randomly selected subject terms for each depth. Notice that all times are less than 0.22 seconds. Figure 8.12(b) shows the corresponding average result sizes.

Fig. 8.12 (Continued)



The same experiments were run on a larger database that does not fit in main memory. The database schema is shown in Table 8.3. Tables `subjectHierarchy` and `subjectHierarchy2` form a balanced and complete tree with degree 10 and depth 5. Again, each paper is associated with one randomly selected leaf subject term from each of the two hierarchies and with 1 to 4 randomly selected authors. All fields of the tables have been indexed with B-tree access method and the size of the database is 1.24 GB (the indexes occupy 718 MB). The experiments run on the same machine as described above but with the shared buffers parameter of postgresSQL set to 1 GB. Figure 8.12(c) shows the measured times (t_a , t_b and t_c). The reported times are the average times of 40 different runs for 10 randomly selected subject terms for each depth. Notice that if the depth of the subject term is greater than 1 then the times are less than 1.5 seconds. Times were gathered using Java, meaning that the overhead of the JDBC driver is also included. Figure 8.12(d) shows the corresponding average result sizes (for depth = 1 we get 200 thousands papers).

A more detailed description of the SQL queries as well as a thorough experimental evaluation is available at [310].

Table 8.3 Schema of larger synthetic database

Relation	Attribute	Number of tuples
paper	<u>pid</u>	2×10^6
	title	
	year	
	venue	
author	<u>pid</u>	5×10^5
	<u>authorName</u>	
paperAuthors	<u>pid</u>	5×10^6
	<u>authorId</u>	
subjectHierarchy	<u>stId</u>	111.111
	name	
	parentId	
paperSubjects	<u>stId</u>	2×10^6
	<u>pid</u>	
subjectHierarchy2	<u>stId</u>	111.111
	name	
	parentId	
paperSubjects2	<u>stId</u>	2×10^6
	<u>pid</u>	

A variation of the previous approach is to use an ORDBMS (Object-Relational Database Management System). This choice has reduced storage space requirements and better performance in some cases, especially if the complete descriptions of objects are stored. Specifically, the set-valued attributes that are supported by ORDBMS, allow having an inverted file-like database representation: for each object we can have only one tuple containing a cell with the identity of the object and a set-valued attribute storing the identifiers of all terms that have been assigned to the object. Other variations are also possible. For instance, we can have a hybrid approach where some of the data are always kept in main memory. For instance, the hierarchically organized attributes values (i.e. the taxonomies) can be kept in main memory while the rest in the DBMS.

8.3 Case Studies: Existing Systems

This section describes in brief architectural issues and existing systems. Figure 8.13(a) shows a general component diagram using the UML notation. It depicts a component named Explorer that provides an interface (called IExplore) offering methods for setting up the focus and for computing and getting the zoom

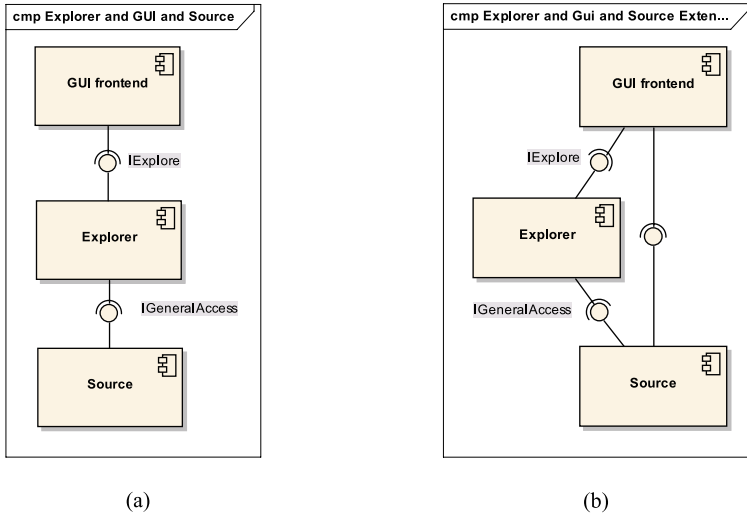


Fig. 8.13 Component diagrams

points as well as other useful information (e.g. the count information for each zoom point). That component (and through the `IExplore` interface) is used by a `GUIfrontend` component that carries out the dialog with the end user (e.g. through one or more tree view graphical components).

The component `Explorer` is fed through an interface, named `IGeneralAccess`. The methods of that interface depend on the application. For instance, the component `Source` can be a plain file system. Alternatively, it can be an inverted file, a DBMS that can be accessed through an ODBC interface, or even a remote source accessed through a number of Web services.

The way the interface `IGeneralAccess` is used (while the user interacts with the system) depends on the adopted architecture. For instance, one architectural choice is to load all metadata (i.e. entire materialized faceted taxonomy M) once and at the beginning. A limitation of this architecture is that all data should fit in main memory. Alternatively, in a partial loading approach, only the restricted materialized faceted taxonomy is loaded. Specifically, the part needed for describing a given set of objects A ($A \subseteq Obj$), i.e. $(\mathcal{F}, I)|_A$ (as defined in Sect. 2.7). This approach is appropriate for cases where M is too big to fit in memory (e.g. the metadata index of a Web Search Engine). Regarding implementation either the *lazy* or the *labor-intensive* (Sect. 8.1.4) strategy can be adopted. Once a restricted (on a set of objects A) materialized faceted taxonomy is loaded using the labor intensive strategy, the component `Explorer` offers all services for restricting the focus without having to call any method of the `IGeneralAccess` interface unless (a) the user wants to use an alternative access method (e.g. to submit a free text query), or (b) the user requests a zoom-out/side operation that can lead to a set of objects A' that is not subset of A .

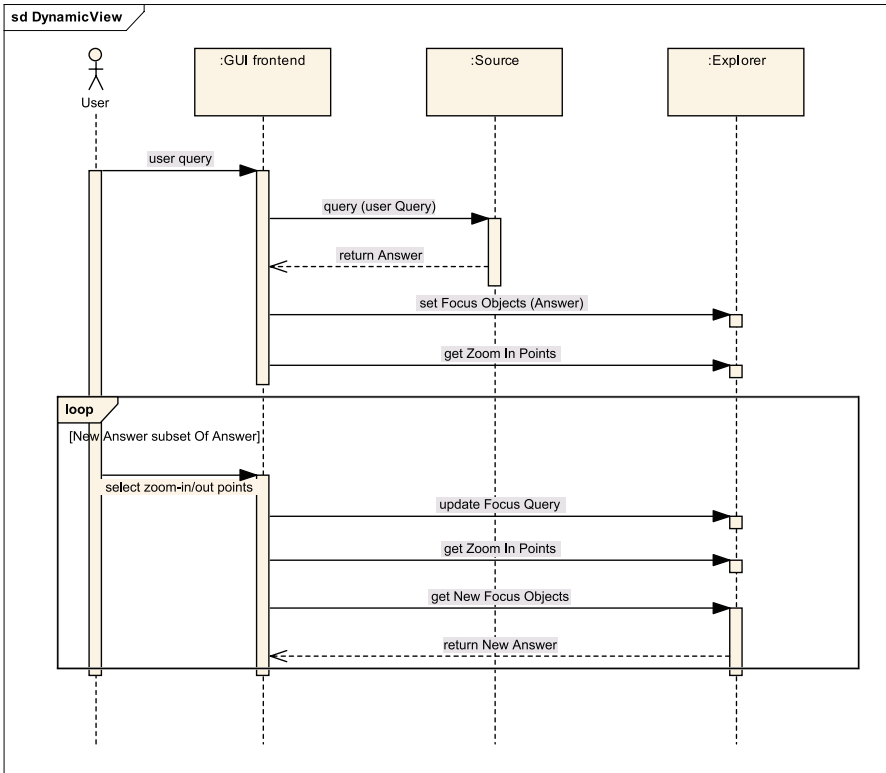


Fig. 8.14 An indicative sequence diagram

Figure 8.13(b) shows how these components are articulated in case both access methods (guided navigation and general access methods) have to be provided through an integrated user interface. In this case, the GUIfrontend component can send directly queries to the underlying source.

Figure 8.14 shows a Sequence Diagram, depicting an indicative sequence of messages, for the case where both guided exploration and general access methods are provided through an integrated user interface that adopts the *labor-intensive* strategy. The loop is continued as long as the requested *NewAnswer* is a subset of the loaded *Answer*. If it is not, then *Explorer* has to be fed again. For instance, if the user selects to zoom-out, and this operation yields to a focus ctx' whose extension is not subset of the current, i.e. $\bar{I}(ctx') \not\subseteq A$, then $setFocusObject(\bar{I}(ctx'))$ has to be issued. The same is true if the user submits a new query q' .

There are several search engines that support (up to some degree), the interaction paradigm of exploratory search. Table 8.4 lists a few representative systems. For each one of them we list some sites in which they are used, plus some other features. Subsequently (in Sects. 8.3.1 and 8.3.2), two systems are described in brief.

Table 8.4 Faceted metadata search engines in commercial sites

Metadata search engines	Used in commercial sites	Support of zoom-in points	Over DBMS	Support of remote sources	I/O formats	Other supported features
Knowledge processors	Non commercial: tiziano.di.unito.it, erare.di.unito.it	Yes	No	Yes		rule-base autoclassifier for XML sources, integrated IR component
CAMELIS	Personal data	Yes	No	Yes (URLs)	CSV, JPEG, MP3, BIBTEX	automatic and manual classification, querying by examples, export of playlists and slideshows
FleXplorer	Non commercial: mitos search engine	Yes	No	Yes	TXT, XML, ...	
i411	ElectionsOntario.on.ca, DeTelefoongids.nl, iLocal.net	Yes	Yes	Yes	HTML, XML, PDF, DOC, PPT, XLS, ...	Predefined taxonomies and categories, compressed on-disk storage
Mercado	Blockbuster.com, Sears.com, USOPNet.com, officemax.com	Yes	Yes	Yes		
Siderean seamark	Indiana Educational Clearinghouse, Fortunoff.com, EnvironmentalHealth-News.org	Yes	Yes	Yes	XML, RDF, RSS, flat files	Predefined taxonomies and categories, uses RDF an intermediate storage format
Endeca	TowerRecords.com, BarnesAndNoble.com, Spiegel.com, Cabot-Corp.com	Yes	Yes	Yes	XML and database imports (Oracle, DB2, SQL Server)	Predefined taxonomies and categories

Table 8.4 (Continued)

Metadata search engines	Used in commercial sites	Support of zoom-in points	Over DBMS	Support of remote sources	I/O formats	Other supported features
Solr	Repubblica.it, StubHub.com, Archive.com, Chowhound.com, CNet.com	Yes	Yes	Yes	HTML, OpenOffice, DOC, XLS, PPT, IMAP, RTF, PDF, etc.	Lucene search library
Google base	base.google.com	Yes (but without count information)	No	Yes	PDF, XLS, TXT, HTML, RTF, WPD, ASCII, XML	

8.3.1 *Flexplorer and Mitos*

This section describes an implementation that is coupled with a Web search engine. *Flexplorer* [309] is a main memory API (Application Programmatic Interface) that allows managing (creating, deleting, modifying) terms, taxonomies, facets and object descriptions. It supports both finite and infinite terminologies (e.g. numerically-valued attributes) as well as explicitly and intensionally defined taxonomies. The former can be classification schemes and thesauri, the latter can be hierarchically organized intervals (based on the *cover* relation), etc.

The system is implemented in Java, so the predefined ordering of built-in types (e.g. of `int`, `float`, `String`), as well as the customized ordering defined for user-defined Java classes (e.g. through the `comparable` interface) can be exploited. To allow intensionally defined partially ordered domains, a `partiallyComparable` interface has been defined and can be used by the developer. The framework also supports parametric types.

Regarding user interaction, the framework provides methods for setting the focus and computing all kinds of zoom points. In addition, the framework allows materializing on demand the relationships of a taxonomy, even if they can be inferred as this can speed up the computation of direct narrower terms (the $Nr^{(1)}(t)$ as defined in Sect. 8.2) at the cost of extra main memory space. Regarding deployment, the framework can be used either at the server side or at client side.

As faceted exploration can be combined easily with other access methods (e.g. information retrieval queries, structured queries, or application-specific queries), the user could start interacting not only by selecting some terms (i.e. by specifying a focus), but through a set of objects. To this end below we discuss one such scenario.

Mitos⁵ is a prototype Web search engine [206, 207].⁶ Flexplorer is used by Mitos in order to offer general purpose browsing and exploration services. Currently, five facets are supported. Specifically, and on the basis of the top- L (where L is typically less than 10,000) answer of each submitted query, the following facets/taxonomies are created and offered to users:

- *web domain*, a hierarchy is defined (e.g. *csd.uoc.gr < uoc.gr < gr*),
- *format type* (e.g. pdf, html, doc, etc.), no hierarchy is created in this case,
- *encoding (or natural language)* of a web page (e.g. utf-8, iso-8859-1),
- *dates hierarchy* (a taxonomy of dates is created automatically from the last modified dates of each page in the result set),
- a hierarchy derived by applying an *on-line results clustering* algorithm. In particular, a novel variation of the STC (Suffix Tree Clustering) method [330] is employed which is described in detail in [161]. The algorithm works on the snippets and this can be done in real time for the top-100 documents.⁷

Figure 8.15 shows a sample screendump of the Web-based GUI (for more, see [208]).

The system follows a partial on-demand loading approach. This approach is beneficial in cases where the materialized faceted taxonomy is too big to fit in memory. The shortcoming of this approach (in comparison to the full loading approach) is that the loading time that has to be paid at every submitted query can be high if the query answer is large. Figure 8.16 shows the loading time of the top- L answer for various values of L : from 10^4 to 10^6 . As the loading time depends on the format employed (and the associated parsing costs), the figure reports the loading times for four different formats, namely: (a) JDBC ResultSet, (b) XML, (c) a (proprietary) TXT-based format, and (d) a main memory format, called ResultDocument that is provided by the Flexplorer API.

The loading time reported for each answer size and for each different method is the average time of 10 executions. As one would expect, ResultDocument is the faster (as it is a main memory “format”), followed by the JDBC ResultSet. Parsing the TXT file is much slower and parsing the XML file is the slowest choice. For the experiments we used a Pentium IV 3 GHz with 2 GB RAM with Windows XP.

Figure 8.17 shows the time to compute the zoom-in points (for the aforementioned set of five facets) after the selection of a zoom-in point, with and without count information. We adopt the lazy, taxonomy-based approach. Each reported time is the average time of 20 executions (5 executions for each for the four loading

⁵In Greek mythology, Theseus succeeded to come out of the labyrinth (of Knossos) by wrapping the “Mitos of Ariadne”, a ball of thread given to him by Ariadne which he had unwrapped when entering the maze. Analogously, Mitos is an engine that aims at guiding users in the Web-labyrinth.

⁶©Department of Computer Science of the University of Crete, and FORTH-ICS, <http://google.csd.uoc.gr:8080/mitos/>.

⁷It is worth noticing that the most time consuming subtask is not the clustering itself but the extraction of the “best text” (snippet) from the cached copies of textual contents of the pages.

Information Systems University of Crete [Advanced Search](#)

Results per page

Faceted taxonomies with on-demand clustering results 9290 Results

You can expand your query with: laboratory security

By clustering

- course (11)
- csd (4)
- department (44)
- forth (42)
- ics (65)
- information (59)
- laboratory (30)
- people (6)
- postgraduate (30)
- program (36)
- projects (9)
- science (56)
- studies (38)
- systems (85)
- undergraduate (6)

REST (9190)

By domain

- gr (9290)

By date

- 2008 (847)
- 2007 (3578)
- 2006 (2291)
- 2005 (282)
- 2004 (159)
- 2003 (238)
- 2002 (171)
- 2001 (80)
- 2000 (25)
- 1999 (25)
- 1998 (9)
- 1997 (8)
- Unknown (1577)

File Systems - 0.24300884

File Systems ...
http://www.csd.uoc.gr/~hy345/notes/html/6_files/frame.htm - 1140011751000 - 1kB Cached [ma

CS-557 Secure Systems - 0.14971472

CS 557 Secure Systems ...
<http://www.csd.uoc.gr/~hy557-1192028439000> - 0kB Cached [mark as spam]

Information Systems Laboratory - 0.1325674

Information Systems Laboratory More accessible version ICS ISL ...
http://www.ics.forth.gr/ics/publications/by_name.jsp?Person_ID=7-0-8kB Cached [mark as spam]

Multiple Processor Systems - 0.074789636

Multiple Processor Systems ...
http://www.csd.uoc.gr/~hy345/notes/html/8_files/frame.htm - 1140011922000 - 1kB Cached [ma

Information Systems Laboratory - 0.0700706

Information Systems Laboratory More accessible version ICS ISL null About ISL ...
http://www.ics.forth.gr/ics/publications/by_year.jsp?Year_of_publication=null-0-16kB Cached [p
spam]

Information Systems Laboratory Contact Info - 0.06837212

Information Systems Laboratory Contact Info More accessible version ICS ISL Contact ... In
 Contact Info **Information Systems** Lab Centre for Cultural Informatics Head
<http://www.ics.forth.gr/ics/contact-info.html> - 1197380853000 - 15kB Cached [mark as spam]

FORTH-ICS: Information Systems Laboratory - 0.06042196

in the transition from traditional **information systems** such as **information** retrieval ... adapt
information systems Such **systems** will be characterized by large scale
<http://www.ics.forth.gr/ics/> - 1173087253000 - 17kB Cached [mark as spam]

Computer Science Department :: Information - 0.058162868

University Of Crete ICS FORTH Socrates Erasmus Main Options Problem report ... Compu
 Science Department **Information** Main Page People Studies Announcements Services Hyp
<http://www.csd.uoc.gr/index.jsp?tid=info&sub=&lang=en-0-16kB> Cached [mark as spam]

Fig. 8.15 Application on a web search engine

format options). Clearly, the computation of zoom-in points with count information is more expensive than without count information: in 1 s we can compute the zoom-in points of 240,000 results with count information, while without count information we can compute the zoom-in points of 540,000 results.

8.3.2 FASTAXON

FASTAXON is a system for designing compound taxonomies based on CTCA (that was described in Sect. 6.1). Using the system, the designer first defines a number of facets and then creates and assigns taxonomies to each one of them. Subsequently the designer can formulate a CTCA expression for specifying the valid compound terms. The formulation is done gradually and at any point the designer is able to browse the dynamically generated navigation tree (as described in Sect. 6.1.3) defined by the formulated expression.

Fig. 8.16 Time to load results to Flexplorer

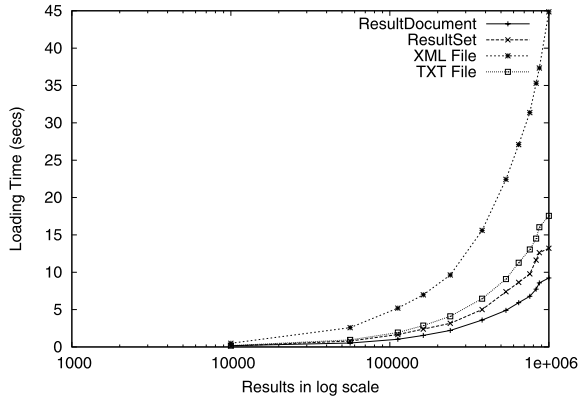
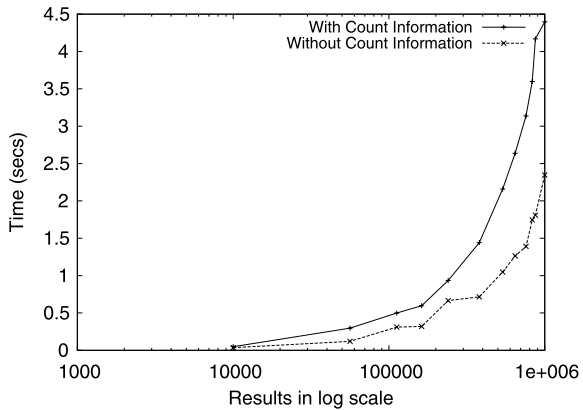


Fig. 8.17 Time to compute zoom-in points using Flexplorer



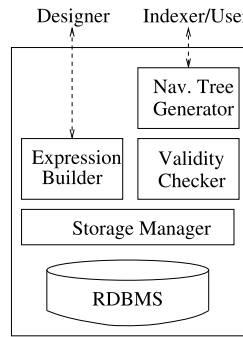
FASTAXON is implemented as a client/server Web-based system written in Java. The server is based on the Apache Web server, the Tomcat application server and uses MySQL for persistent storage. The user interface is based on DHTML (dynamic HTML), JSP (Java Server Pages) and Java Servlet technologies (J2EE). The client only needs a Web browser supporting JavaScripts. Figure 8.18 illustrates the general architecture of the system.

The faceted taxonomy is stored using three tables:

```
FACETS(id:Int, name:String)
TERMS(id:Int, name:String, facetId:Int)
SUBS(termId:Int, broaderTermId:Int)
```

The first stores the names of the facets, the second stores the terms of T , and the third the transitive reduction (Hasse diagram) of the subsumption relation \leq . Clearly, we also have the corresponding foreign key constraints $TERMS.facetId \subseteq FACETS.id$, $SUBS.termId \subseteq TERMS.id$, and $SUBS.broaderTermId \subseteq TERMS.id$.

Fig. 8.18 The architecture of FASTAXON



Concerning the storage of the CTCA parameters we can distinguish two cases. If all elements of the parameters P and N of e contain at most one term from each terminology T_i then we can store all P and N parameters in one table `PARAMS` that has one attribute A_i for each facet F_i , $i = 1, \dots, k$, and $dom(A_i) = T_i$. Specifically, each element $s = \{t_1, \dots, t_m\}$ in P (or N) is stored as a tuple r such that $r(A_{J(t_i)}) = t_i$, for each $i = 1, \dots, m$ where $J(t) = m$ iff $t \in T_m$. To speed up the lookup operations on the relation `PARAMS` (i.e. the selection queries), each operation in the parse tree of the expression e is assigned a unique identifier `opId`. This identifier is associated with each element of the parameter, P or N , of the operation. This has been implemented by adding one additional column `opId` to the relation `PARAMS`. In this way, the elements of each parameter P or N can be collected and searched more efficiently. However, in the general case where there are elements in P or N that contain more than one term from one facet (e.g. when there are one or more self-product operations), we cannot use the table `PARAMS` as defined earlier, because we would need set-valued attributes and the relation would not be in 1NF. For this reason we use a scheme which “simulates” set valued attributes (we assign an identifier to each value set and store the values of the set in a separate table). Specifically, we use two tables:

```
PARAMS(A1, ..., Ak, opId)
SETS(setId: Int, termId: Int)
```

where `SETS.termId` \subseteq `TERMS.id`.

Let s be a compound term that is element of one parameter P or N , which is stored as a tuple r in `PARAMS`. If there is $i = 1, \dots, k$ such that $s \cap T_i = \{t_1, \dots, t_n\}$ with $n > 1$, then we create a new identifier `setId` for $\{t_1, \dots, t_n\}$ and store `setId` in the appropriate attribute of r , i.e. $r(A_i) = setId$. The correspondence between `setId` and the terms $\{t_1, \dots, t_n\}$ is stored in `SETS`.

For example, consider the expression

$$e = (Sports \oplus_{P1} Location) \ominus_N (\oplus_{P2}^* (Facilities))$$

where

$$P1 = \{\{SeaSports, Crete\}, \{WinterSports, Greece\}\}$$

$$P2 = \{\{IndoorSPool, OutdoorSPool\}, \{OutdoorSPool, Jacuzzi\}\}$$

$$N = \{SeaSports, Crete, IndoorSPool\}$$

where *IndoorSPool*, *OutdoorSPool*, and *Jacuzzi* are terms of the facet *Facilities*. The contents of the tables PARAMS and SETS follow:

Sports	Location	Facilities	opId
SeaSports	Crete		o1
WinterSports	Greece		o1
		s1	o2
		s2	o2
SeaSports	Crete	IndoorSPool	o3

setId	termId
s1	IndoorSPool
s1	OutdoorSPool
s2	OutdoorSPool
s2	Jacuzzi

8.4 Formats and Protocols

This section describes some formats that can be used to exchange taxonomies and object descriptions (some of them have already been mentioned in Sect. 3.5).

- **XML** is a meta-language for defining markup. It provides a syntax for structured documents but does not impose any semantic constraint on the meaning of these documents.
- **XML Schema** is a language for restricting the structure of XML documents.
- **RDF** is a data model for objects (resources) and relations between them. It is actually a data model based on object-attribute-value triples. Such triples can be represented and exchanged in various formats including XML (i.e. RDF/XML) and TriG [43].
- **RDF Schema** is a vocabulary description language for describing properties and classes of RDF resources equipped with subsumption semantics. It allows expressing ontologies.
- **OWL** is a richer vocabulary than RDF Schema that allows expressing disjointness and cardinality constraints as well as other properties like equivalence, symmetry, transitivity.

As a sample, we describe XFML, an XML format for publishing and connecting faceted metadata between Web sites.

XFML (eXchangable Faceted Metadata Language) is described as an open XML format for publishing and connecting faceted metadata between Web sites. It is based on the Topic Map [11] standard but uses only a subset of its data representation capabilities. Roughly, XFML allows expressing *topics*, organized in hierarchies (trees) within mutually exclusive containers called *facets*. These structures are called *maps*. The topic concept is the same as it is in Topic Maps, e.g. “love”, “5 o’clock in the afternoon”, “Shakespeare” are potential topics. The notion of topic does not correspond to the notion of term as used in thesauri [149], because a topic is an abstract entity, while a thesaurus term is a specific (set of) words. This means that the topic “accessibility” could be described by the term “accessibility”, but also by the terms “universal access” or “ease of access”. XFML doesn’t deal with terms, it deals with topics, actually with topics organized in specialization/generalization hierarchies (trees). Each topic belongs to one and only one facet and there can’t be two representations of the same topic in a map. A topic can have only one other topic as general topic. In addition, each topic may be associated to a published subject indicator (psi), i.e. to a human readable resource that defines the topic.

Facets are mutually exclusive containers that contain hierarchies of topics. Mutually exclusive means that a certain topic can only possibly belong to one facet. *ThingsToDo* and *PlacesToGo* are good facets, because a topic can never be both a thing to do and a place to go to. *People* and *Colors* are two other good facets. *Cities* and *PlacesToVisit* are “bad” facets if used in the same map because *Pisa* (a potential topic) could belong to both.

Concerning indexing and XFML, a Web page can be indexed with respect to a map, by assigning to it a set of topics from that map.

XFML also allows establishing *connections* between different maps, by indicating that a topic in one map is equivalent to (synonym of) a topic in another map.

XFML allows publishing maps, page indices and connections in an XML based format. The substantial part of the XFML DTD⁸ is shown in Fig. 8.19, while Fig. 8.20 sketches the XFML representation of an example.

Note that the element *occurrence* has also an attribute *strength* (whose range is $[1, \infty]$) that allows indicating how strong the indexer believes that a particular topic is to a particular page. An occurrence strength of 1 means complete confidence while higher numbers mean less confidence. Note that the availability of such estimates motivate supporting best match (as opposed to exact match) retrieval models over taxonomy-based sources. Such models are not described in this book, however Sect. 5.6 discussed some of the rising issues.

In addition, XFML specification includes some processing instructions for the applications⁹ which describe how one application can exploit a connection to a map B in order to import the object indices of map B or to browse map B. There is already software for XFML, including tools for creating and managing XFML maps, XFML parsers and converters from XFML to XTM [11] and RDF [6].

⁸XFML DTD can be found in <http://www.xfml.org/spec/xfml.dtd>.

⁹See <http://www.xfml.org/spec/1.0.html>.

```

<!ELEMENT xfml (mapInfo | facet | topic | page)*>
<!-- Facets -->
<!ELEMENT facet {#PCDATA}> <!ATTLIST facet id ID #REQUIRED>
<!-- Topics -->
<!ELEMENT topic (name, connect*, psi*, description?)>
<!ATTLIST topic
  id ID #REQUIRED
  facetid IDREF #REQUIRED
  parentTopicid IDREF #IMPLIED
>
<!-- Pages -->
<!ELEMENT page (title?, description?, occurrence*)>
<!ATTLIST page url CDATA #REQUIRED>
<!ELEMENT occurrence EMPTY>
<!ATTLIST occurrence
  topicid IDREF #REQUIRED
  strength CDATA #IMPLIED
>

```

Fig. 8.19 The DTD of XFML

```

<xfml>
  <!-- FACETS -->
  <facet id="places">places</facet>
  <facet id="sports">sports</facet>
  <topic id="earth" facetid="places">
    <name>Earth</name>
  </topic>
  <topic id="greece" facetid="places" parentTopicid="earth">
    <name>Greece</name>
  </topic>
  <topic id="finland" facetid="places" parentTopicid="earth">
    <name>Finland</name>
  </topic>
  ...
  ...
  <!-- PAGES -->
  <page url="http://www.greekhotels.com/AgapiBeach">
    <occurrence topicid="crete" />
    <occurrence topicid="windsurfing"/>
  </page>
</xfml>

```

Fig. 8.20 Example of XFML

Finally, RDF is clearly a format that can be used as dynamic taxonomies need only two kinds of relations (`hasType` and `subclassOf`). However there is not native notion of facet.

Regarding *protocols*, there is not a particular protocol dedicated to the needs of exploratory search. However the design of such a protocol could be done quite straightforwardly, and implementations over different technologies (e.g. Web Services) could come up easily.

As an example we could mention JSR-170¹⁰ which is a type of Object Database tailored to the storage, searching, and retrieval of hierarchical data. The JCR API grew out of the needs of content management systems, which require storage of documents and other binary objects with associated metadata. In addition to object storage the JCR provides APIs for versioning of data, transactions, observation of changes in data, and import or export of data to XML in a standard way. An open source implementation of JCR API is the Apache Jackrabbit JCR.¹¹ It is not hard to see that one could easily extend such APIs in order to enable exploratory search, i.e. to add methods for computing and returning zoom-in points.

8.5 Composition of Taxonomies with Logic Components

Several information processing domains have components in which logic plays a crucial role: e.g., information systems and information retrieval [259, 312], logic-based diagnosis [214], logic-based programming [173, 180]. In particular, using logics in information systems combining querying and navigation is the starting point of Logical Information Systems (LIS) [101, 104]. Logics can also be used in dynamic taxonomies, as shown in Sect. 5.4. Taxonomies are particular cases of logics, and logics can be used to automatically generate taxonomies. Some advantages of logics over usual taxonomies are a tighter combination of expressive querying and flexible navigation, and the automatic extraction of a navigation vocabulary out of large or infinite sets of concepts.

We are interested in logics as well-defined formalisms for representation and reasoning. We start with concrete data-types such as integers or strings, e.g., for the annotation of photos. Then, in order to extend the expressivity of descriptions and queries, we want to add intervals on integers, patterns on strings (e.g., “contains”), structures such as tuples or sets, boolean operators, etc. We do not claim that assembling these features will lead to universal logics, but the other way round, universal logics such as first-order logic do not provide such features, and cannot be extended easily. Decidability and tractability is another constraint that prevents us from using universal logics, especially if their universality is not required in practice.

Moreover, we do not want to fix the logic *a priori*, say to some description logic [57], as we want the application designers to be able to *customize* it. Indeed, no single logic will fit all needs, as different applications require different concrete domains and different trade-offs between expressivity and efficiency of the subsumption test. A second requirement on these logics is that they should be *embeddable*, so that building an application only requires plugging a logic in a generic information system. The problem of this approach is that defining a logic (i.e., its syntax, its semantics, its decision procedure and correction proofs) requires logic expertise, which application designers are unlikely to have. A work that shares our motivations

¹⁰<http://en.wikipedia.org/wiki/JSR-170>.

¹¹Release 1.0, April 2006.

for the customization and embeddability of logics is LeanTAP [33]. Their authors propose a style of theorem proving that is so concise that it is very easy to modify it in order to accommodate a different logic. However they have overlooked the fact that the average user is not a logic expert, and we think that few people could effectively customize LeanTAP to their needs in practice. Moreover, if somebody manages to perform such a customization, there is no guarantee that the result will be a consistent and complete prover.

We present a toolbox of *logic components* that helps in constructing new logics, i.e., define their syntax and semantics, and automatically derive their decision procedure and proofs about their consistency/completeness. We formally introduce *logic functors* as logic components. Each logic functor applies to logics, and returns a new composed logic: e.g., the product or the sum of two logics, the boolean closure of a logic (connectors `and`, `or`, `not`), concrete domains like integers or strings. Composing a logic is then similar to composing an algebraic expression from basic operations: e.g., given two logics L_1 and L_2 , and a binary functor F , $F(L_1, L_2)$ is a new logic with sub-logics L_1 and L_2 . Every logic functor is equipped with a partial decision procedure (as well as other procedures), and given the expression of a composition of logic functors the *Composer* automatically produces a full decision procedure (as well as other procedures). Every logic functor is also given a set of partial proofs (its type) that expresses its behavior with regard to consistency/completeness, so that checking and proving the properties of composed logics can be done automatically by the *Composer*.

In Sect. 8.5.1, we introduce the notions of syntax, semantics, procedures, and properties that make up a logic. Section 8.5.2 defines logic functors and their composition, which includes the production of decision procedures and their type-checking. A few logic functors are shortly presented (a full description of them and others is available in a research report [105]), and a complete implementation of them is also available.¹² The effectiveness of our approach is demonstrated in Sect. 8.5.3 by the definition of a logic combining various taxonomies and concrete domains, and in Sect. 8.5.4 by the reconstruction of the description logic \mathcal{ALC} with logic functors only.

8.5.1 Logics

We are interested in logics for representing concrete values and data structures, and querying such representations with patterns like intervals, substrings, and boolean combinations of such patterns. This leads us to define logics in a very practical and generic way, in the form of an *abstract data type*. This definition is an extension of Definition 5.1. The notions of formulas, subsumption and abstraction have already been discussed in Sect. 5.4.

¹²LogFun: a library of logic functors (<http://www.irisa.fr/LIS/ferre/logfun>).

Definition 8.1 (Logic) A logic L is a tuple (L, S, P, T) , where:

- L is the *abstract syntax*, i.e., a set of formulas;
- S is a model theoretic *semantics*, i.e., a domain of interpretations I and a satisfaction relation $(\models) \subseteq I \times L$ between interpretations and formulas;
- P is a set of logical *procedures*:
 - $(\sqsubseteq) \subseteq L \times L$, a decision procedure for subsumption,
 - $\top \in L$, the most general formula,
 - $\perp \in L$, the most specific formula,
 - $(\sqcap) \in L \times L \rightarrow L$, the conjunction operation,
 - $(\sqcup) \in L \times L \rightarrow L$, the disjunction operation,
 - $abstr \in L \rightarrow \mathcal{P}(L)$, the abstraction operation (see Definition 5.4);
- T is the *type*, i.e., a set of properties (and their proofs) about the behavior of procedures with regard to semantics, e.g.:
 - \sqsubseteq -consistency: for all $f, g \in L$, $f \sqsubseteq g \implies (\forall i \in I : i \models f \implies i \models g)$,
 - \sqsubseteq -completeness: for all $f, g \in L$, $(\forall i \in I : i \models f \implies i \models g) \implies f \sqsubseteq g$,
 - $abstr$ -consistency: for all $f \in L$, $\forall g \in abstr(f) : f \sqsubseteq g$.

The abstract syntax is left totally free, and can mix connectives and values such as strings or numbers. Procedures are free to implement any algorithm, so that only proved properties can guarantee they have their expected behavior. For instance, the decision procedure \sqsubseteq for subsumption is correct with regard to semantics iff consistency and completeness are satisfied. If consistency is not satisfied, there may be false positives, and if completeness is not satisfied, there may be false negatives in subsumption tests. Similar properties are defined for other procedures. The abstraction operation is used to extract a taxonomy from the classification of a set of objects by logical formulas. It is applied to every object descriptor, and the results are joined to make up the taxonomy set of concepts. A property to be satisfied by $abstr(f)$ is that every generated formula subsumes f .

Logics are implemented as modules. They implement a same *signature* \mathbb{L} that acts as an embeddability contract between generic systems and logics. This signature is composed of:

- a type for the internal representation of formulas (abstract syntax),
- a parser and a printer for the concrete syntax,
- a function for each procedure, e.g., the decision procedure is a function that takes two formulas as arguments, and returns true or false,
- a function that returns for each property, either a proof of it, or an explanation why no proof could be completed.

An information system relies on this signature to build a taxonomy (extracting a finite set of concepts and computing subsumption relations between them), and to compute extensions of concepts (computing subsumption relations between object concepts and query concepts). Section 5.4.2 explains how logics are used in dynamic taxonomies.

8.5.2 Logic Functors

Suppose we have several logics that we want to combine so as to form a more complex logic. We need an operation that takes one or several logics and returns a composed logic. Such an operation is called a *logic functor* and is simply defined as a function from logics to logics. We extend the definition of logic functors to 0-ary functors, i.e., to logic components that need no argument to form a logic. Then composing a logic consists in applying a logic functor to one or several sub-logics, which can be either 0-ary functors, or again composed logics. This composition process is illustrated in the following subsections.

For instance, the functor $Prop(X)$ is the propositional logic (logical connectors and, or, not), whose atoms have been abstracted by the formal parameter X , and can thus be replaced by more complex formulas. The decision procedure of propositional logic has been adapted to make use of the logical procedures of X . For instance, given that a, b, c are X -formulas, we have a and $b \sqsubseteq c$ iff $a \sqsubseteq_X c$ or $b \sqsubseteq_X c$ or $a \sqcap_X b \sqsubseteq_X c$. The proofs of consistency and completeness of this decision procedure are also parameterized by property proofs in X . For instance, the subsumption of $Prop(X)$ is consistent iff the procedures $\sqsubseteq_X, \perp_X, \sqcup_X$ are consistent and the procedures \top_X, \sqcap_X are complete. The same applies for all components of a logic, and for each logic functor.

Logics being implemented as modules, logic functors are implemented as *parameterized modules* [174], which correspond to generics in Java, templates in C++, and functors in Objective Caml. This implementation allows to directly use the programming language compiler for composing logic functors and logics, so that we did not have to write a specific logic composer.

8.5.3 Combining Attributes, Concrete Domains, and Taxonomies

In this section, we show how a complex logic can be obtained from simple and reusable components. These components are either atomic logics (0-ary functors) that generally correspond to concrete domains, or n -ary functors for combining or extending logics. In the following, when defining a logic functor, we only define its abstract syntax, interpretation domain, the subsumption procedure, and the abstraction procedure. Indeed, syntax, subsumption and abstraction are what matters from the point of view of dynamic taxonomies, and the interpretation domain clarifies the meaning of formulas. Other procedures and properties are accessible in full details in a research report [105].

To start with, we assume we have the following atomic logics (see Sect. 5.4 for details and examples):

- `Location`: taxonomy of locations, as a logic,
- `String`: logic of strings and substrings,

- Integer: logic of integers and intervals,
- Date: logic of dates and intervals of dates.

Logics `Integer` and `Date` could possibly be decomposed further as both manipulate intervals.

These logics are generally used as monodimensional taxonomies, while this is not required. For instance, a photo is usually given one location, one date, one size, and one comment. We need to combine these logics to form a multidimensional logic, where each object can be given at the same time a location, a date, a size, and possibly several comments. To this purpose, the logic functor `Sum` can be used as it produces the union of two logics. Its effect on taxonomies is to put them side by side.

Definition 8.2 (Functor `Sum`) The functor `Sum` takes two logics L_1 and L_2 , and returns their union:

- $L = L_1 \cup L_2$,
- $I = I_1 \cup I_2$,
- $(\sqsubseteq) = (\sqsubseteq_1) \cup (\sqsubseteq_2)$,
- $abstr = abstr_1 \cup abstr_2$.

Then a common root is added by applying another logic functor `Top`, whose only effect is to add a most general formula that subsumes all formulas.

Definition 8.3 (Functor `Top`) The functor `Top` takes a logic L_1 , and extends it with a top (most general formula):

- $L = L_1 \cup \{thing\}$,
- $I = I_1$,
- $(\sqsubseteq) = (\sqsubseteq_1) \cup (L \times \{thing\})$,
- $abstr(f) = abstr_1(f) \cup \{thing\}$, for all $f \in L_1$.

The logic allowing to describe photos with 4 kinds of properties, as mentioned above, is defined as

$$L1 = \text{Top}(\text{Sum}(\text{Location}, \text{Sum}(\text{String}, \text{Sum}(\text{Integer}, \text{Date}))))$$

In some applications, locations may be used under different roles. For instance we may want to distinguish between where some person lives *in*, and where she comes *from*. The composite logic `Top(Sum(Location, Location))` is unsatisfying because (1) it duplicates the taxonomy of locations, and (2) elements from the 2 copies cannot be distinguished. Instead we introduce a taxonomy of roles as a logic `Role`, which contains at least the roles *in*, *from*, and a top role *any* meaning “any role”. Then we combine this logic and `Location` with the logic functor `Prod`, which produces the product of two logics.

$$L2 = \text{Prod}(\text{Role}, \text{Location})$$

The formulas of the product logic are pairs, and two pairs are ordered by subsumption iff the first and second parts are respectively ordered.

Definition 8.4 (Functor `Prod`) The functor `Prod` takes two logics L_1 and L_2 , and returns their product:

- $L = L_1 \times L_2$ (formulas are pairs),
- $I = I_1 \times I_2$ (interpretations are also pairs),
- $(f_1, f_2) \sqsubseteq (g_1, g_2) \iff f_1 \sqsubseteq_1 g_1 \wedge f_2 \sqsubseteq_2 g_2$, for all $(f_1, f_2), (g_1, g_2) \in L$,
- $abstr((f_1, f_2)) = abstr_1(f_1) \times abstr_2(f_2)$, for all $(f_1, f_2) \in L$.

For instance, a person living in Paris and coming from Spain is described by `in Paris` and `from Spain` (to be read “from somewhere in Spain”). The abstractions of `from Spain` are `from Spain`, `from Europe`, `from somewhere`, `any Spain`, `any Europe`, and `any somewhere`. The following subsumption relations are verified:

- `from Spain` \sqsubseteq `from Europe` \sqsubseteq `any Europe`,
- `from Spain` \sqsubseteq `any Spain` \sqsubseteq `any Europe`.

We emphasize the fact that these relations are automatically deduced from the relations between roles on one hand, and between locations on the other hand. If a new location, say Italy, is inserted in the taxonomy of locations, all pairings of a role with Italy become virtually present in L_2 . The same happens for the insertion of a new role. This results in an expressive logic for describing and querying data, while requiring no additional effort from the information system designer.

The logic L_2 can replace the use of `Location` in logic L_1 or, even better, the notion of role can be generalized to all kinds of concrete domain logics, resulting in the logic

$$L_3 = \text{Top}(\text{Prod}(\text{Role}, \text{Sum}(\text{Location}, \text{Sum}(\text{String}, \text{Sum}(\text{Integer}, \text{Date}))))))$$

In this logic the formulas, i.e. concepts, can be seen as valued attributes over various concrete domains. Additional concrete domains can then easily be added to this logic definition.

8.5.4 Reconstructing the Description Logic \mathcal{ALC}

Description Logics (DL)¹³ are widely used in knowledge representation and information systems [57]. Like our logics, they are based on a notion of subsumption \sqsubseteq . So, an interesting question is to know whether logic functors can be used to reconstruct description logics. We show in this section that the subsumption test of \mathcal{ALC} (without terminological axioms) can be easily reconstructed. \mathcal{ALC} is the subset of OWL DL restricted to the constructors $\top, \perp, \neg, \sqcap, \sqcup, \exists, \forall$.

For recall, the abstract syntax of the logic \mathcal{ALC} is defined by:

$$C \rightarrow A \mid \top \mid \perp \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \exists r.C \mid \forall r.C$$

¹³See Sect. 3.5.1.

where A and r respectively stand for *concept names* and *role names*. These are properly represented by logical atoms (logic functor Atom).

Definition 8.5 (Functor Atom) The functor Atom takes no argument logic, and simply defines a set of incomparable atomic formulas (i.e., names):

- L is a set A of atoms,
- $I = A$,
- $f \sqsubseteq g \iff f = g$, for all $f, g \in L$,
- $\text{abstr}(f) = \{f\}$.

In the semantics of \mathcal{ALC} a formula $\forall r.C$ is equivalent to the formula $\neg\exists r.\neg C$, so that we can restrict ourselves to the existential quantification $\exists r.C$. When some object satisfies a formula $\exists r.C$, this means it is related to another object that is instance of C , through a relation satisfying r . So the property $\exists r.C$ can be represented as the couple (r, C) of a role, and a complex concept (logic functor Prod): its concrete syntax can easily be customized as $\exists r.C$ for readability. All other connectors of the language are provided by the logic functor Prop , shortly introduced above.

Definition 8.6 (Functor Prop) The functor Prop takes a logic L_1 , and returns its *propositional closure*:

- L is the smallest set of formulas containing L_1 , the most general formula \top , the most specific formula \perp , and such that for every formulas $f, g \in L$, the formulas $\neg f$, $f \sqcap g$ and $f \sqcup g$ also belong to L ,
- $I = I_1$,
- the decision procedure \sqsubseteq is based on existing provers for the propositional logic [33], extended to make use of the decision procedure of L_1 (see [105] for details),
- for all $f \in L$,

$$\text{abstr}(f) = \begin{cases} \text{abstr}_1(f) & \text{if } f \in L_1, \\ \text{abstr}(g) \cup \text{abstr}(h) & \text{if } f = g \sqcap h, \\ \{f\} & \text{otherwise} \end{cases}$$

This leads to the following first attempt of defining the logic \mathcal{ALC} with logic functors.

$$\text{ALC} = \text{Prop}(\text{Sum}(\text{Atom}, \text{Prod}(\text{Atom}, \text{ALC})))$$

The Composer produces a new logic ALC from its defining expression. Note the way it is recursively defined in order to account for complex concepts inside quantifiers. The Composer also performs its type-checking, whose result says the subsumption is proved consistent, but not complete.

The above definition is correct with regard to syntax, but what about the interpretation domain? According to the definition of functors (see above), it is recursively defined as

$$I = A \cup (A \times I)$$

i.e., an interpretation is either a primitive concept (an atom), or a pair made of a primitive role (an atom), and another interpretation. This is not satisfactory because an \mathcal{ALC} -interpretation should not be a single atomic concept or single role, but should be a set of atomic concepts and roles. This adaptation to the interpretation domain can be obtained by the logic functor Set , whose interpretation domain is the power set of the interpretation domain of its argument logic.

Definition 8.7 (Functor Set) The functor Set takes a logic L_1 , and returns a logic with the same abstract syntax, but whose interpretation domain is the power set:

- $L = L_1$,
- $I = \mathcal{P}(I_1)$,
- $(\sqsubseteq) = (\sqsubseteq_1)$,
- $\text{abstr} = \text{abstr}_1$.

By applying the functor Set on the argument of the functor Prop , we obtain the new definition

$$\text{ALC} = \text{Prop}(\text{Set}(\text{Sum}(\text{Atom}, \text{Prod}(\text{Atom}, \text{ALC}))))$$

This time the interpretation domain is $I = \mathcal{P}(A \cup (A \times I))$, and the resulting subsumption prover is proved both consistent and complete by the Composer. The formula of ALC

$$d(o) = \text{Tall} \sqcap \exists \text{child}.\text{Male} \sqcap \forall \text{child}.\text{Tall}$$

can be used to describe someone (object o) who is tall, has a male child, and has only tall children. The formula

$$q = \exists \text{child}.\text{(Male} \sqcap \text{Tall)} \sqcap \forall \text{child}.\text{(}\neg \text{Male} \sqcup \text{Tall)}$$

can be used as a query to get people who have a tall male child, and whose male children are all tall. It can be proved in ALC that $d(o) \sqsubseteq_{\text{ALC}} q$ holds, hence $o \in \text{objects}(q)$.

8.5.5 Conclusion

Logic functors provide an effective framework for the “engineering of logics” [27] that makes it possible for an end-user to design safely a new logic that is tailored to a specific application. A decision procedure for the subsumption test and its consistency/completeness proofs are derived automatically. If necessary the proofs exhibit prerequisites that may indicate how to build more consistent/complete variants of the logic. The framework is implemented in Objective Caml, and uses in a crucial way the capability of the ML family to develop parameterized modules (also called functors in that domain). It could as well be implemented in object-oriented languages that support parameterized classes (generics in Java, and templates in C++).

The framework occupies an original position between plain programming that allows for arbitrary expressivity but in which almost nothing semantic can be proved automatically, and option selection that is very easy to use but leaves very little choice [84]. This original position offers a trade-off between expressivity and ease of use, where logics as expressive as \mathcal{ALC} can be designed and built by application designers with no logic expertise.

A toolbox of logic functors has been developed and collected into a library, LOG-FUN.¹⁴ It is compatible with a generic implementation of dynamic taxonomies, CAMELIS,¹⁵ which is implemented along the lines of the base implementation of dynamic taxonomies, except that it relies on logics rather than taxonomies. Given an application domain, a logic can be composed from logic functors, and then plugged in CAMELIS. This results in an application-specific software.

¹⁴<http://www.irisa.fr/LIS/ferre/logfun/>.

¹⁵<http://www.irisa.fr/LIS/ferre/camelis/>.

Chapter 9

Applications and Experiences

Giovanni Maria Sacco and Sébastien Ferré

“By far the best proof is experience.”

Sir Francis Bacon, 1561–1626

This chapter discusses a number of real-world applications of dynamic taxonomies. Most current applications are object-seeking or knowledge-seeking exploratory tasks, and address important areas such as e-commerce, multimedia infobases, diagnostic systems, digital libraries and news systems, e-government, file systems, and geographical information systems. Applications in these areas are discussed in detail in the following, and applications in cultural heritage, art and architecture, e-recruitment, e-hrm, e-matchmaking, e-health, and e-learning are briefly reviewed.

9.1 Introduction

The application range of dynamic taxonomies is extremely wide and, in fact, most tasks which are traditionally considered as search tasks are actually exploratory tasks and can benefit from the superior exploration capabilities of dynamic taxonomies. As we mentioned in Chapter 1, there are basically three tasks in which exploration is important:

1. pragmatic *find the right object* tasks in which the user has to find the object which best fits his requirements. This is by far the most frequent task and the best example is product selection in e-commerce;
2. *knowledge-seeking* tasks which feature an exploration similar to the one offered by hypertext systems: the user attention is not focused on a specific requirement, but he is following his interests in a free way, in order to increase his knowledge on the topics described by the infobase. An example is access to an encyclopedia with topics organized as dynamic taxonomies;
3. *wisdom-seeking* tasks, in which the goal is understanding the inner laws of the corpus, in order to derive a deeper knowledge of the corpus itself. Data mining extensions, which enable the use of dynamic taxonomies for this type of applications, were discussed in Sect. 5.1. Although the vast potentialities of this approach are still to be tapped, an example may be given by web log analysis, where the exploration capabilities of dynamic taxonomies allow to analyze web logs according to user interests and discover in this way trends which are actually interesting and statistically significant.

Most current exploratory tasks fall in the pragmatic *find the right object* paradigm. This class and the *knowledge-seeking* class include an extremely large number of different applications which are, at least in part, reviewed in the following. We will specifically describe

- e-commerce
- multimedia infobases
- diagnostic systems
- digital libraries and news systems
- e-government
- file systems
- geographical information systems and
- general-purpose Web search engines¹

These topics are by no means exhaustive. Additional interesting application areas are:

- cultural heritage, art and architecture. Applications in this area tend to follow the knowledge-seeking paradigm. One of the best-known applications is probably Flamenco's faceted search access to the San Francisco Fine Arts Museum image collection [134, 327]. A museum application is also discussed in [147], and a more recent application managing architectural data within the MACE project is reported by Stefaner et al. [280].
- e-recruitment, e-hrm, e-matchmaking. The object-seeking exploration paradigm which applies to product selection also applies to the selection of persons, in two important application areas: e-recruitment [5, 244] and e-hrm (electronic human resource management) [38, 39] in general and their consumer-to-consumer counterpart, match-making [233]. Both areas are high-growth Internet sectors, and both benefit from the exploration capabilities provided by dynamic taxonomies. The full integration of text retrieval allows to represent only common features through the taxonomy, and use text retrieval to query for other features, so that relatively simple taxonomies can be effectively used even for general applications [244];
- e-health. The term e-health denotes healthcare practice supported by electronic processes and media, usually through Internet. Applications in this area include web-based diagnostic systems [243, 254] (reviewed below), medical guidelines and literature [318], and general health portals [146];
- e-learning. E-learning systems have recently attracted a large interest because they allow high-quality remote, asynchronous education which supports rich material such a text, video, and offer a high level of involvement of teachers and students by providing collaborative tools, such as forums. In this context, dynamic taxonomies can play an important role for teachers and students alike. They provide a powerful exploration system for the large content bases of e-learning environments, which makes locating and reusing relevant material a simple task, and

¹Described in Sect. 8.3.1.

at the same time encourages strong interdisciplinarity by making different aspects of a same subject (discussed in different courses) immediately available. Demo and Angius [86] describe the integration of dynamic taxonomies in Moodle [193], a course management system for cooperative learning.

9.2 E-commerce

E-commerce portals have been one of the Internet areas experiencing the fastest growth since the beginning. There are compelling and obvious economical reasons behind this growth: low overhead, just-in-time supply, and the widespread acceptance of online credit card payments. The economics of small or no inventories is a compelling force towards very large stores: successful stores such as amazon.com have shifted from a focused line of products to selling quite diverse and heterogeneous items.

Although product selection is *the* critical point in an e-store, this was a most frustrating experience in the early days, as traditional search technologies such as database queries, information retrieval or primitive exploration techniques such as hypermedia were used. Since customers do not know exactly the specific object they want, but are rather looking for the object which best fits their individual requirements, system assistance in browsing and exploration is required.

Dynamic taxonomies have been applied to e-commerce [231–233, 237] since 1999 [220], with a prototype system for the selection of digital cameras.² Other systems, such as Endeca [98] and Mercado [188],³ were introduced a few years later. However, the single event which prompted the exponential growth of DT applications in e-commerce, was probably their adoption by Yahoo! Shopping [322] in late 2004.

E-commerce and e-auctions are perhaps the best examples of a *find the right object* exploratory tasks [239, 245, 250]. Most users of e-commerce portals do not look for a specific product but want to find the ‘right’ product in a possibly quite large set of alternative products. The right product depends on how product features are evaluated according to user requirements (perceptions, interests, financial capabilities, etc.). Different users (and, most importantly, the same user at different times) are likely to weigh each feature differently, and consequently select different products. It is unlikely that users are able to associate a precise numeric weight to each feature, but they can easily rank features in decreasing order of importance. So, in addition to a primary interest focus (e.g., budget price), users will have a secondary, tertiary, etc. focus: e.g., budget cameras with the highest resolution vs. the lightest budget cameras available. A secondary focus depends on the user preferences but also on the features which items in the primary focus exhibit, and so on.

²This is historically the first application of dynamic taxonomies to e-commerce; an updated version is available at [219], and it is shown in Fig. 9.1.

³Other systems are reviewed in Sect. 8.3.

We split the interaction into two stages in cascade: the thinning-game and the end game. In the *thinning game*, the user is confronted with a large number of items and has to reduce it to a small set of candidates to be further inspected. The thinning game works on a potentially quite large amount of data, and should assist the user by providing fast and effective thinning of the product infobase on the base of selected features. Dynamic taxonomies, which offer systematic and guided thinning, represent the current solution of choice for the thinning game.

In the *end game*, a single product to be purchased has to be selected from the set of candidate products, by comparing their features. Although the total number of features to be compared might be quite large, the end game involves a small number of products, and a small amount of data as compared to the thinning game. The problem here is not data management, but rather a data presentation problem, i.e., how to present data in order to assist the user to discriminate among different products.

9.2.1 The Thinning Game

The product selection process is exemplified by a digital camera shop, with examples in Fig. 9.1.

In order to thin the number of alternatives the user has to:

1. find all the available features,
2. focus on the most relevant one for him (the primary focus), which discards all the products without that feature,
3. find all the features for the products retained,
4. select the next focus among them, and iterate the process until the number of candidates is sufficiently small.

The major critical point is the conceptual summary of related features, i.e., the reduced taxonomy which allows the user to easily find the features (e.g., resolution, zoom) for his focus (say, cameras under \$200) and zoom on one or more of them. This discards the products which do not have those features and consequently thins candidates out. If it is not available, the next focus cannot be set and the thinning game is already over: the user has to inspect all the inexpensive cameras and find their features by manual inspection.

Other important points in the thinning game are the ability to operate on products at a set-at-a-time rather than at an instance-at-a-time level (the primary focus defines a set of products, a secondary focus intersects the primary focus set with the set defined by the secondary focus, etc.), and to have systematic summaries of sets (the current focus) in real time. Finally, since the number of features for large stores can be quite large, a taxonomic organization of features is usually required. As we remarked, product presentation tends to be a second-order concern in the thinning game.

When the user enters the digital camera shop, he finds a concise systematic summary of the products (Fig. 9.1a). The numbers before each feature indicate how

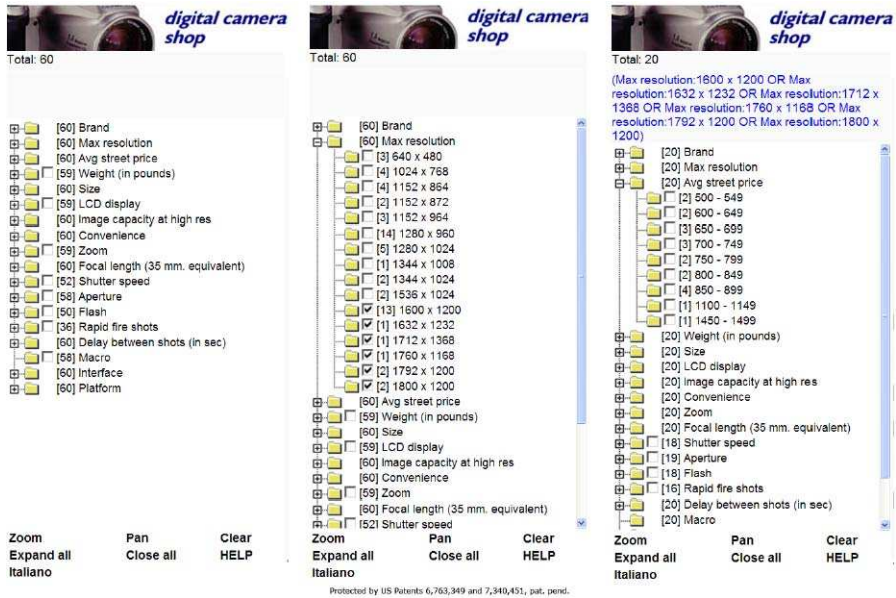


Fig. 9.1 From left: a Initial taxonomy for digicam shop; b Preparing to zoom on high-resolution digicams; c Reduced taxonomy after zoom on high-resolution digicams

many products have that feature (i.e., are classified under it). Assume the user is interested in high-resolution (1600 × 1200 or more) digital cameras. With a conventional taxonomy, he would retrieve 20 cameras, and he would need to inspect each camera in order to select the right one. Instead, he performs a zoom on high-resolution cameras (Fig. 9.1b). The zoom operation focuses on the cameras he just selected (20 out of 60) and shows a concise summary (Fig. 9.1c) of all the features for only and all such cameras. Here, he expands Size, but he could explore any other relevant feature (Price, Shutter speed, etc.). If he is interested in Pocket cameras, he does not need to perform another Zoom operation because only 3 cameras are selected and he may reasonably look at them, i.e., he is now entering the “end game”.

A single zoom operation reduced the number of candidate cameras from 60 to 3 (the ones classified under Pocket cameras), with a total freedom to express requirements and a complete assistance from the system in selecting interesting features. All the requirements for the thinning game are satisfied. The user is effectively guided to reach his goal by a clear listing of all possible alternatives and the guidance given by conceptual summaries makes product selection quicker and more accurate. Quick product selection is not just a psychological impression, but derives from the superior scalability and quick convergence of dynamic taxonomies.⁴

In addition, and most importantly, users feel they have considered all the alternatives in reaching a result, and they clearly understand how the result was reached. By

⁴See also Sect. 3.2.

contrast, text retrieval is usually perceived as a game of chance, while agent technologies [269] are not transparent and usually cannot explain how the result was obtained to the casual, untrained user.

As we mentioned in Chap. 1, the product selection problem can be stated as a classical optimization problem: the ‘right’ camera is the one which minimizes a weighted combination of features. From this point of view, the thinning game implemented by dynamic taxonomies is a heuristic strategy and can consequently be suboptimal. There are two major shortcomings. First, boolean cutoffs (accept/reject) for feature values are used. As an example, a boolean cutoff at a price of \$200 (i.e., price no higher than \$200) rejects both a potentially interesting product at \$205 and a clearly too expensive product at \$1,000.⁵ Second, a non-compensatory strategy is used: once an object is rejected because of any feature, that object is eliminated from further consideration, even if some other features might compensate its local suboptimality. For these reasons, the thinning game does not necessarily provide the best deal for the user.

Optimal solutions can be obtained by the *weighted additive* (WADD) strategy, which computes a weight for each object as the average weight of feature values for that object; feature values are weighted over each feature, and features are weighted among them [210]. The “best deal” is the object with the highest weight. WADD is a compensatory strategy and allows high scores on one feature to compensate for low scores on another feature.

Although optimal in theory, it is unlikely that strategies like WADD can produce optimal results in practice, for the following reasons:

- optimal results in WADD depend on accurate weights, and it is unlikely that casual users are able to supply them and, most importantly, understand the impact of different weights on the actual result.⁶ In dynamic taxonomies, the user is only required to select, at each stage, the most important feature values, a much easier task;
- in WADD, the number of weights to be supplied by the user is quite large in practice and most of them are probably useless. In dynamic taxonomies, instead, only the first most important features need to be selected, and, because of the quick convergence to small candidate sets, the number of selections is very small (2–3 in most practical situations);
- in WADD, an agent is required to derive the final solution, and it is usually difficult to clearly explain to the user how the answer was found. In dynamic taxonomies, it is the user himself who explores the information base according to his interests and the effects of his operations are clearly understood.

In conclusion, even if the thinning game might be suboptimal and direct the user towards products which are not the “best deal”, it is fast, concrete and easily un-

⁵This problem can be solved by extending dynamic taxonomies by using fuzzy [329] rather than boolean logic. See also Sect. 5.6.

⁶Qualitatively specified preferences are more expressive than quantitatively specified preferences [66, 109].

derstood by users, so that it is probably the most effective practical solution to the problem.

Other benefits of dynamic taxonomies for the thinning game over other approaches include [245]:

- easy support of multilingual access, which is more and more important because of the transnational nature of most e-commerce sites;
- transparent and unobtrusive gathering of user preferences, by simply monitoring the concepts used for zooming. This can be used for personalization and improvement of user experience in a much more reliable than traditional techniques [25], and also provides a more reliable and complete base for data mining and automatic recommendation applications [265];
- advanced features such as user reviews or object popularity can be easily accommodated by specific facets and fully integrated in the exploration process. As an example, a facet for popularity can be used to easily select the most popular products among candidates, e.g., the most popular cameras among high-resolution 10× zoom cameras. Global popularity indicators, by converse, are not really useful: a user interested into high-resolution 10× zoom cameras would not care to know that the most popular camera is a zoomless budget model.

9.2.2 *The End Game*

The second stage, the *end game*, is entered when the user has arrived at a suitably small set of candidate items and must now select the single object to purchase, by comparing features of candidate items. Although a growing number of e-commerce portals let users compare different products, such functionality is usually primitive and only supports the side-by-side display of features. A typical example is shown in Fig. 9.2, which compares three Nikon digicams. Alternating gray and white backgrounds are used to increase row readability: however, the user has no orientation in the comparison of features, so that selecting the “right” camera requires the comparison of all the features.

Simple as this mechanism is, it poses significant cognitive problems because there are usually many features to consider and the number of candidate items is often larger than ten. Most practical situations may require hundreds of comparisons, but even the comparison of two items can be difficult: not only all the features must be compared, but all the different features must also be remembered. Different features will be stored in the user short-term memory [190] which holds 7 ± 2 items. Comparing more than nine feature values becomes therefore quite taxing and usually leads to user disorientation so that users will need additional tools, such as pencil and paper.

In order to simplify the end game and reduce the number of comparisons the user must perform, the user should be assisted in quickly finding *discriminants* among different items, i.e., features with different values which can guide the selection. At the very minimum, features whose values are the same over all the items, and



Fig. 9.2 Comparison of features for Nikon digicams

are therefore useless as discriminants, should be quickly perceived as such, and discarded on demand.

In addition, the user selects the final object by informally “weighing” the desirability of a combination of features of interest. In many practical cases, values of specific features can be ranked a priori from the less desirable to the most desirable value, using an intuitive *ceteris paribus*⁷ semantics [128]. For instance, all the other features being equal, a smaller price is always better than a higher one. These rankings can be used in such a way that the user quickly perceives the desirability of feature values in a row, instead of comparing them exhaustively.

Figure 9.3C shows the *enhanced feature display* [237, 238, 245], according to the requirements discussed above. Features with the same value over all the products are identified by a gray background and can be hidden. For most of the features (e.g., price, resolution), values are ranked according to their desirability: the color of the background goes from a bright green (for the best values) to a bright red (for the worst values); a white background is used for mean values. As an example, prices higher than the mean price are red (the higher the price, the brighter the color) and prices lower than the mean price are green. Some features, such as brand or body color, cannot be ranked, so that they are not color-coded and their background is white.

⁷Literally, *everything else being equal*.

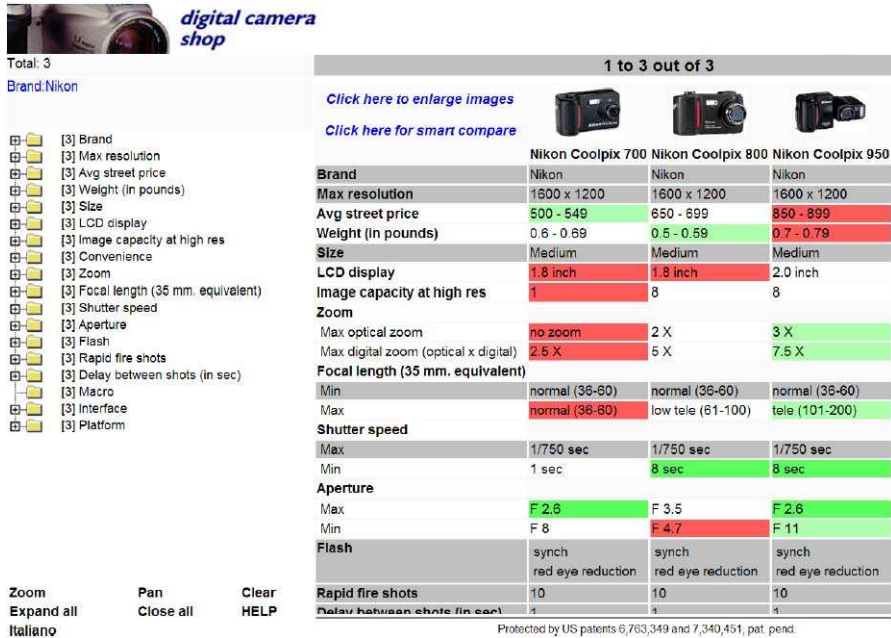


Fig. 9.3 Enhanced feature display for Nikon digicams

Color-coding shows quickly and clearly what’s different and what’s not, but also indicates where products differ more dramatically: a preliminary assessment can be usually done at a glance. In Fig. 9.3, the relative ranking of the three Nikon cameras is immediate and simply based on the number of red and green features of each camera, whereas it requires a careful inspection of all the features in Fig. 9.2. Several variations on color-coding are discussed in [245].

The end-game can be further improved by highlighting skyline [46] products, i.e., products which are not *dominated* by any other product according to a set of features. For example, if price and optical zooms are the only features we consider, product A with a \$400 price and a 3× optical zoom is dominated by product B with a \$200 price and a 6× zoom, because product B is better (less expensive and a with a better zoom) than product A. Non-dominated products are worse deals with respect to skyline products and can be ignored.⁸ Skyline queries can be based on “*ceteris paribus*” criteria, and are especially interesting here because they are applied to a small number of items, and therefore require negligible computational resources.

Although we have separated the thinning-game and the end-game into two different phases, a tighter coupling between them might be desirable. Sacco [237] proposes to use the color-coded grid as the single interface for product selection and to integrate dynamic taxonomy feature selection and summaries in it.

⁸We show them in an appropriate way rather than ignore them, because users can be influenced by incommensurable features, such as brands.

9.3 Multimedia Information Bases

Very large multimedia information bases are a rapidly expanding and pervasive reality thanks to advances in storage and network technologies: examples range from music to video-on-demand applications, to large photo banks. In this section, we are primarily interested into user-centric access to image infobases, but our approach [240, 241, 256] can be extended to generic multimedia databases.

Current research focuses on two different and unreconciled approaches for accessing multimedia databases: the metadata approach as opposed to the content-based approach. In the metadata approach, each image is described by metadata. Metadata types range from a set of keywords (tags, as in Flickr [111]) or a textual description, to standardized structures for metadata attributes and their relationships like the MPEG-7 standard [182], to ontologies [266]. Some metadata (e.g., image format) can be automatically derived from the image itself, but the vast majority are manually entered. Once images are described by metadata, their actual contents becomes irrelevant for browsing and retrieval so that it is straightforward to use techniques such as database queries on structured metadata or text retrieval queries on metadata consisting of a textual description of the multimedia object.

The content-based approach (CBIR, content-based image retrieval) describes the image through low-level multimedia features (e.g., color [119], texture [177], shape [151]) automatically extracted from images. Retrieval is based on similarity among images: the user provides an image (selected through metadata or, in some cases, at random) and requests similar ones.

Both approaches suffer from significant drawbacks. Semantic annotation is costly, especially for large, existing databases. In addition, the metadata approach relies on descriptions which are known to be inaccurate, heavily dependent on the specific human classifier, ambiguous, etc. These problems can be alleviated by using ontological metadata rather than plain text descriptions, but a level of subjectivity remains. The CBIR approach is based on the automatic extraction of “descriptions” from the image itself. This approach seems to solve all the problems because extraction is inexpensive and image descriptions are objective, without the inconsistencies caused by human classification. However, despite significant improvements, the accuracy of CBIR systems is still less than satisfactory, because of “the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for the user in a given situation” [272]. No CBIR system will be able to reconstruct all relevant information in all situations: in many practical cases, the information is just not there [256].⁹

The dichotomy between the two approaches (metadata vs. primitive features) is a major flaw in current image retrieval, because it forces the user to use different access strategies which only depend on the type of feature (conceptual or low level multimedia feature) he is considering. Most importantly, from the user point

⁹For example, it is usually impossible to derive the location of a photograph from low-level multimedia features.

of view, none of these approaches really supports an exploratory access to the image collection, which we believe to be the most common access paradigm in these applications. Exploration is not an additional, desirable feature of a multimedia information retrieval system. On the contrary, in most practical cases, retrieval without exploration is just a trial-and-error task, with no guarantee of the quality of the user's final choice.

The access paradigm supported by most current image retrieval systems is quite different. Systems based on metadata use access techniques such as queries on structured data or text retrieval techniques which do not support exploration. CBIR systems use information retrieval techniques which are centered on retrieval by similarity. This type of access affords a very simple and intuitive user interaction, but offers no clue on what the information base contains. Systems which organize images through hierarchical clustering do offer an initial systematic summary of the collection, but do not account for the iterative refinement required by our working definition of exploration. From this point of view, they are similar to traditional taxonomies which offer conceptual but static summaries. In addition, as we remarked in Sect. 3.1.2.1, hierarchical clustering does not produce IS-A taxonomies.

There are very few exceptions. A dynamic taxonomy approach for multimedia metadata was proposed by Sacco [236] and used by Hearst et al. [134] in their Flamenco prototype system, which was successfully applied to a rather large image collection [327]. These works rely on conceptual metadata features and ignore primitive multimedia features.

From another perspective, El Niño, a prototype system by Santini and Jain [261], focuses on browsing (rather than exploration) through low level multimedia features and textual descriptions. El Niño works on a multi-feature weighted similarity measure and relies on a visual relevance feedback interface to modify these weights and try to match the user notion of similarity. The system has some conceptual similarities with the Scatter-Gather text retrieval system [76], discussed in Chap. 3.

The approach described here [240, 241, 256] proposes a significant change in access paradigms, based on dynamic taxonomies. A working system (Rosso Tiziano, available on-line at [287]) was implemented with several goals:

- to provide a single, coherent framework which seamlessly integrates access by metadata and access by low level multimedia features for querying. This framework considerably simplifies user access, and each access method reinforces the effectiveness of the other one;
- to support the exploration of image collections, so that both metadata and primitive features can be used to express interest foci, and at the same time to systematically summarize them, in order to guide the user towards his goal;
- to provide a test bed for the evaluation of different strategies, integrated access and human factors in general, in which different primitive features approaches can be easily integrated and compared.

9.3.1 Combining Conceptual Access with Low Level Multimedia Features

Access by primitive multimedia features is usually based on clustering: objects are grouped on the basis of the values of one or more features, say color, according to a measure of similarity between any two objects. Image clustering is conceptually similar to clustering for textual documents.¹⁰

For all access purposes, how images are clustered is immaterial. A cluster merely identifies a set of objects which are grouped together by some affinity. This definition is equivalent to the definition of a concept in a dynamic taxonomy: a concept denotes a set of objects classified under it, rather than a set of properties which instances of a concept must satisfy. This equivalence between concepts and clusters suggests that clusters can be integrated in a dynamic taxonomy by adding a top-most concept (or ‘facet’) for each clustering scheme, with its actual clusters being the sons of this concept.

For instance, if a clustering scheme by dominant color exists, a facet labeled “dominant color” will be added at the top-most level. Its sons will be all the clusters grouping objects by dominant color similarity. For large information bases, where a very large number of clusters is to be expected, hierarchical clustering can greatly simplify user orientation and the effectiveness of interaction.

One of our major goals is to achieve an easy and complete integration between access by metadata and access by low level multimedia features. The benefits of such an integration are essentially two. First, combinations of concepts supply a conceptual context, and consequently reduce noise. Such a context, as Santini and Jain remarked [261] “is essential for determining the meaning of an image and for judging image similarity”. Second, when the user starts from low level multimedia features, dynamic taxonomies can be used to quickly summarize the result according to the original conceptual taxonomy, thus increasing the precision of the result and allowing to quickly correlate low level multimedia features with metadata.

As an example, consider a user starting from a low level multimedia feature such as a blue dominant. Focusing on it, the system will show in the conceptual summary that images with a blue dominant include skies, sea, lakes, cars, etc. Conversely, a user focusing on skies will find that images of skies may have a blue dominant, but also a yellow one, a red one, etc. In both cases, the conceptual summary indicates which conceptual contexts are available to further refine user access.

The integration of metadata access with access by low level multimedia features boosts the effectiveness of each type of access. Very simple features, such as dominant color, are probably not sufficient per se to provide an effective access to a large image base: however, when combined with other (low level or conceptual) features, they can provide a useful discrimination. On the other hand, the availability of low level features may reduce the number of metadata to be manually entered, and even

¹⁰Reviewed in Chap. 3.

a simple and small metadata structure can produce significant benefits in access and exploration because of the extremely fast convergence of dynamic taxonomies.¹¹

Although our primary focus is the integration of low level multimedia features and metadata, the dynamic taxonomy approach can be used in a number of variations, by considering metadata only or primitive features only, or by integrating traditional CBIR retrieval by similarity with a dynamic taxonomy metadata description in order to clarify contexts for similar objects. Finally, our approach is neutral with respect to classification and easily accommodates automatic concept recognition from visual features of images [81], an emerging and extremely promising field in image retrieval.

9.3.2 *Monodimensional vs. Multidimensional Clustering for Low Level Features*

Multimedia objects can be described by a growing number [81] of different low level multimedia features such as color, texture, etc. Most current research computes an overall similarity distance for the images in the database by a weighted combination of the similarity of each individual feature. In fact, CBIR systems only support access by similarity and consequently must have a single measure of similarity even when several features are used.

Clustering groups objects together according to a single similarity measure, and consequently each object belongs to one and only one cluster. We call this type of clustering a *monodimensional clustering* by analogy with classification theory, where a monodimensional classification scheme classifies each object under one and only one concept. If a hierarchical clustering scheme is used, monodimensional clustering is equivalent to a traditional, monodimensional taxonomy.

In alternative, we can consider each feature independently for clustering: each feature will result, in general, into a different clustering scheme because, for instance, two objects similar by texture may have different colors. In this case, an object will belong to different clusters. We call this approach *multidimensional clustering* because it is similar to a multidimensional classification scheme, where an object o is classified under different concepts.¹²

We now compare monodimensional clustering schemes to multidimensional clustering schemes. It is immaterial whether features are primitive multimedia features (e.g., color) or conceptual features, such as a painter name. It also does not matter whether some features are in fact derived from other features: for instance, color which can be characterized by dominant or average. All the available features

¹¹See Sect. 3.2.

¹²In order to avoid confusion, please note that classic clustering theory is indeed defined in a multidimensional space: the difference being that in classic clustering an object o only belongs to single cluster, whereas clustering schemes based on multidimensional classification can place the same object in different clusters, according to different similarity measures.

can be represented by independent clusters, and a single object o will belong to F clusters. By switching from monodimensional clustering to multidimensional clustering on F features, the cost of clustering increases by a factor F because all the objects have to be fully clustered for each feature.

We criticize monodimensional clustering on two major points:

1. the notion of similarity, which we contend is inaccurate and ineffective in monodimensional clustering and classification, and
2. the exponential growth in the number of clusters which is required if we want to keep the number of objects in a cluster at a reasonable level.

In monodimensional clustering, a given multimedia object is represented by a point in a multidimensional space, computed as the weighted combination of the object's primitive multimedia features. Similarity between any two objects is a function of their distance and depends on the weights used to combine low level features. This single and fixed set of weights cannot be satisfactory in every situation. Consider two features such as color and texture: user A may be more interested in color than in texture, whereas user B could be more interested in texture than in color. This implies different weights on these features and consequently different similarity functions, so that objects a and b can be similar for user A and quite different for user B.

However, since a single, predefined similarity function is used for clustering, the resulting clustering scheme only accommodates those users whose notion of similarity matches the predefined function. The only way to accommodate differences among users, would be to perform clustering dynamically on a similarity function given by the user himself. However, this is not feasible for two reasons:

1. cost, as the dynamic reclustering of large information bases requires substantial resources and time. Reclustering is required in El Niño [261]. A similar approach in textual databases, Scatter–Gather [76], was criticized in Sect. 4.6.9;
2. human factors, because it is unlikely that the average, unskilled user would be able to understand the effects of weights and hence come up with appropriate weights for different features. A similar problem occurs in product selection in e-commerce, discussed above.

The effects of monodimensional clustering can also seriously bias experimental results, because it may be difficult to discriminate between the actual advantages offered by specific features and the random effects of a specific weighting scheme.

With respect to convergence, a hierarchical monodimensional clustering scheme is analogous to a traditional monodimensional taxonomy. The analysis reported in Chap. 3 applies, and indicates that multidimensional clustering schemes have a better scalability, a dramatically faster convergence to small result sets and the ability to correlate different features. In addition, custom similarities can be easily expressed. As an example, users can zoom on a specific texture cluster, and immediately see all the color clusters for that texture.

In summary, we believe that multidimensional clustering strategies deserve close attention,¹³ because of their faster convergence and, most importantly, because they present visual similarities according to different perspectives (color, luminance, etc.), and allow for a more articulated exploration of the information base.

9.3.3 Representing Low Level Multimedia Features

The information base used as an example in this section consists of 251 images of five masters of the Italian Renaissance: Piero della Francesca, Masaccio, Antonello da Messina, Paolo Uccello and Raphael. Each work was thoroughly classified according to a number of topics which include, among others, the painter name, the type of painting (single, polyptic, etc.), the technique used (oil, tempera, etc.), the period in which it was painted, current locations, the themes (religious painting, portrait), etc. Differently from other test image databases, which usually exhibit widely different images, the images in the sample collection are relatively similar and therefore harder to characterize by low level multimedia features. In addition, the collection is a good representative of one of the important applications of image retrieval: museum and art collections.

In addition to metadata, each image in the sample collection was automatically described by a number of independent primitive features. These include:

- average image brightness
- average image saturation
- HSV histogram
- clustering on average color in CIE L·a·b color space on a 4×4 grid

Each image was first reduced to a 250×250 size, while preserving the aspect ratio. For all the features except the last, images were converted to the HSV color space, which separates brightness, saturation and hue, because it is perceptually more accurate than the RGB color space, and because brightness and saturation are important descriptive features.

Rather than an average image hue, the image color histogram was recorded, using the color reduction technique proposed by Zhang et al. [333]. A Gaussian blur is applied before conversion to the HSV color space, in order to avoid color noise. Finally, HSV colors are mapped to 36 bins based on the perceptual characteristics of the color itself.

The last primitive feature records average colors. Each image axis is subdivided into 4 intervals, giving 16 areas: the average color of each area is computed. The aspect ratio is not preserved. In order to compute average colors, the image is first converted to the CIE L·a·b color space, because linear color combinations in this

¹³Multidimensional clustering is used also by the FCA-based approach of Amato and Meghini [20].

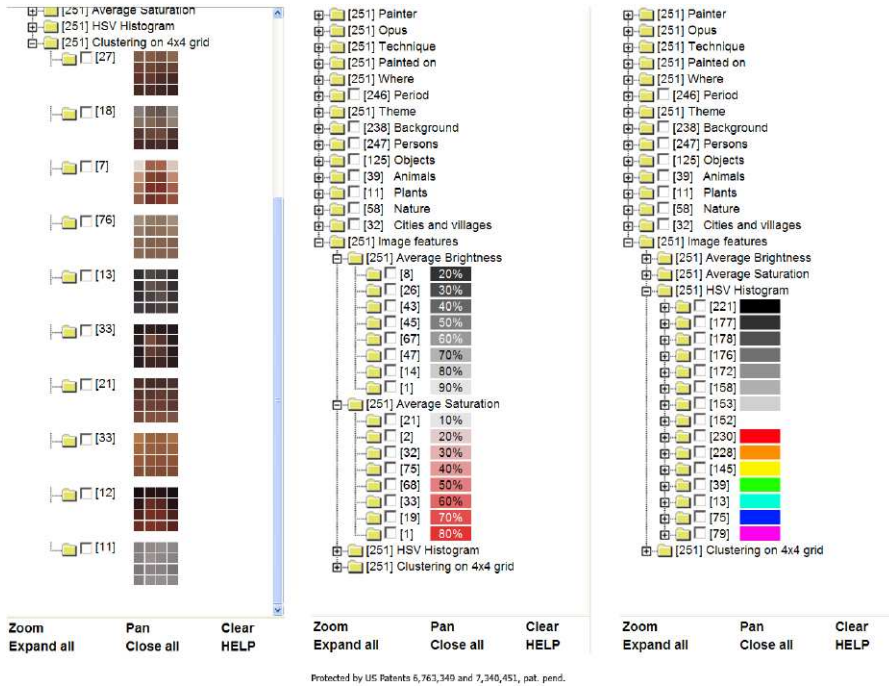


Fig. 9.4 From left: **a** Multidimensional primitive features: clustering of average color on a 4×4 grid. Clusters are labeled by their barycenter; **b** Monodimensional primitive features: average brightness and average saturation; **c** Bidimensional primitive features: reduced HSV histogram

space are perceptually more accurate than in other spaces. Each image is then represented by a vector on 16 dimensions and clustering is applied, based on the cosine measure. The entire collection was clustered into 10 clusters, by using an agglomerative complete-link clustering algorithm, which minimizes the maximum pairwise distance among elements in the cluster [152]. We predefined 10 clusters, a number chosen according to the guidelines from [236]: a larger collection would require a hierarchical clustering scheme. This organization is shown in Fig. 9.4Ca.

When clustering is required, the problem of conveying the “meaning” of each cluster by appropriate labels has to be solved. In traditional clustering, clusters are usually labeled by their centroid: either the cluster barycenter or the object closest to it. Even with text-based clustering, the meaning of the cluster is often so unclear as to pose serious cognitive challenges to the user [236, 260]. With image representations, these problems are considerably worse. As an example, assume that a *tondo* painting¹⁴ is used to label a cluster produced by brightness clustering. Since tondos are rather uncommon, users are likely to assume that the cluster contains tondos, whatever the correct interpretation might be. In order to avoid these problems, we

¹⁴A painting which is circular instead of rectangular.

chose to label clusters by their stylized barycenter rather than by the image closest to the barycenter.

Although the discussion above focused on clustering, many primitive features can be represented by a priori fixed hierarchical organizations of the feature space. As an example, monodimensional image data such as average or dominant color, overall luminance, etc., can be easily represented by a facet with k sons, each representing a possible value. The label of each son can usually be provided by the value (color, luminance level, etc.) itself. In the example, we subdivided both average brightness and saturation into 10 intervals (Fig. 9.4Cb). Although such a subdivision is rather gross, a finer subdivision would probably be useless from the perceptual point of view.

A similar organization can be used for a number of bidimensional features, such as color histograms [110]. The color histogram facet can be represented by a 2-level hierarchy, where colors are enumerated on the first level (immediate sons of the facet). Each color is further characterized, at the next level, by its normalized count, organized as range of values. The structure reported in the figures is slightly more complex than normal histograms. Here, in fact, the highest level reports principal colors (8 shades of gray and the 8 main colors, red to purple); at the lower level, each main color is subdivided into 4 actual colors, by using different luminance and saturation (Fig. 9.4Cc).

9.3.4 Examples of Exploration

Figures 9.5–9.7 report three different explorative sessions which show the significant advantages of the current approach. In the first session, the user starts her exploration from a primitive feature, the average image brightness, and selects dark paintings, i.e., paintings with a brightness of 20% or less. After the zoom operation, the reduced taxonomy in Fig. 9.5C indicates that only Antonello da Messina and Raphael painted dark paintings, and that almost all such paintings are portraits with a black background. The explosion of the Technique topic shows that both painters are the only masters in the collection who use oil painting, rather than *tempera*.¹⁵

In the second session, exploration starts from metadata: after a zoom on Painter > Masaccio, followed by a zoom on Theme > Sacred, the result is summarized according to the HSV histogram, and paintings which have an orange-ish color are displayed (Fig. 9.6C). Almost all the sacred paintings by Masaccio fall in this category: in fact, a golden background (the so-called *fondo oro*) is typical of Italian sacred paintings of the fifteenth century.

Finally, the clustering on a 4×4 grid facet is used to summarize and explore the Portraits by Antonello da Messina. Most portraits fall in a single cluster, which means they are visually very similar. In fact, almost all the portraits by Antonello

¹⁵Tempera uses water and egg-yolk (instead of oil) as a binding medium. Tempera paintings tends to be much lighter.

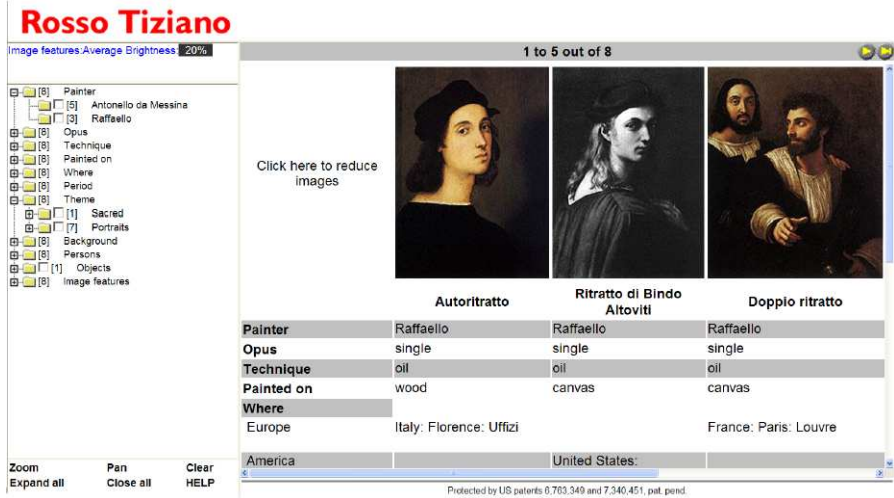


Fig. 9.5 Exploring dark paintings: only Raphael and Antonello have dark items, and almost all are portraits. Dark portraits are expanded

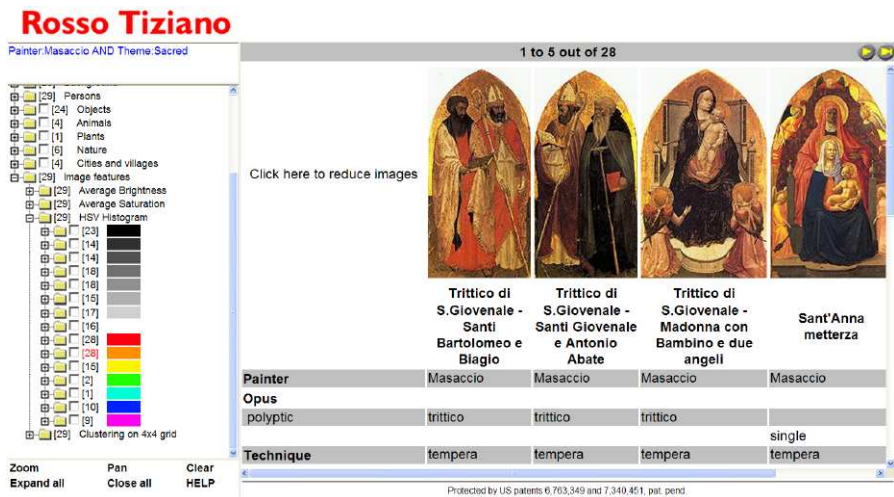


Fig. 9.6 Histogram summary of Masaccio’s sacred paintings: paintings with orange-ish colors are displayed

have a very dark background (see the first session) and the face covers most of the painting (Fig. 9.7C).

These sessions show how the information base can be effortlessly explored, gaining insights on its contents: in fact, we discovered relationships and features of the image collection which no other access method would have made available.

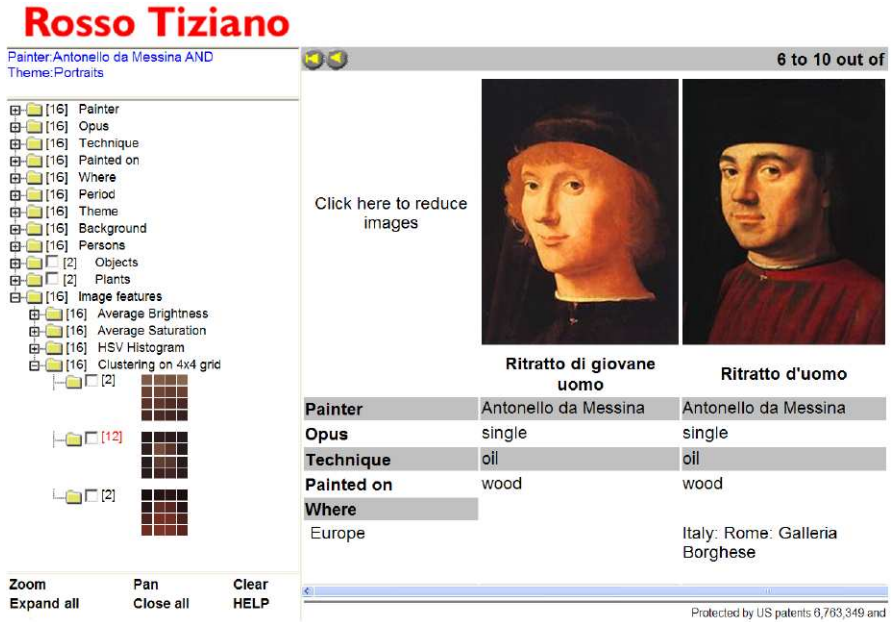


Fig. 9.7 Cluster summary of Antonello’s portraits: displaying the selected cluster

Although formal usability studies are required, the first informal tests on people reasonably familiar with the paintings in the information base show some interesting trends. First, most explorations start from metadata with primitive features used in later stages in order to find visual similarities among paintings characterized by semantic metadata descriptions. If confirmed, this would support our claim that both access by metadata and by primitive features are required and must be dealt with in a uniform way. In addition, low-level-feature-only CBIR’s which do not support metadata access would not seem to match user interactions and requirements.

Second, users found that the ability to see images according to different and independent visual features quite important in exploring the information base, and in discovering effective visual similarities. Again, if confirmed, this would make a case for multidimensional clustering and simple, easy-to-understand primitive features. Traditional CBIR systems strive to capture image “semantics” through a mix of several high quality features. From our initial findings, it seems much more important that the features used are easily understood by users and that they capture image characteristics which are useful for exploration. Because of the very quick convergence of dynamic taxonomies, we claim that an array of simple features, such as the ones we used, may be adequate for exploration.¹⁶

¹⁶This approach is called RAIF (redundant array of inexpensive features) in [256] by analogy with RAID techniques in disk technology. In both cases, the intelligent combination of very simple

9.4 Diagnostic Systems

Medical diagnosis is becoming increasingly difficult because of the emergence of pathologies which are infrequent, either because they are not locally endemic or because they exhibit a small number of cases over the entire world. General practitioners are very often required to diagnose and correctly treat diseases that they are not familiar with and might have not even heard of. In this context, but also with more frequent pathologies, computer assistance can be an essential tool in health-care, especially if it also supports all relevant information such as protocols, best practices, emergency alerts, etc.

Current systems are not successful because they do not involve the physician in the diagnostic process. Dynamic taxonomies can be adapted to the problem of interactive diagnosis and provide a paradigm which places the practitioner at the center of the decision process. Medical diagnosis is seen as the exploration and thinning out of candidate pathologies on the basis of clinical signs and other features. Early experiences with an interactive web-based freely available tool for the diagnosis of rare diseases are presented.

9.4.1 *Computer-Assisted Medical Diagnosis*

Computerized medical diagnosis has a relatively long history and several theories of diagnosis were proposed, with an evolution from systems based on empirical classification rules [54] to theoretical and model-based approaches, such as set-covering [227], abductive diagnosis [73] and consistency-based diagnosis [82]. However, traditional systems have not been successful, for the same reasons which affect most knowledge-based systems [48]. The principal limitation is their system-centric rather than user-centric approach: a master–slave relationship in which the system is in charge of diagnosis, and the user is basically an input device which supplies the system with observations. This dramatically limits their application in areas such as healthcare, where highly skilled professional users do not readily accept such a slave–master relationship. In addition, these systems tend to be oracles: they offer a diagnosis, but are usually unable to explain why in a way users can understand. Finally, the cost of building and maintaining these systems is often so large as to be impractical.

As an alternative, the diagnostic problem can be seen in terms of information access. Complex knowledge-based architectures can be replaced by a collection of (possibly standardized) electronic texts which describe pathologies, which can be searched by traditional techniques such as information retrieval. The creation and maintenance of the knowledge base is greatly simplified, but the limitations of information retrieval are well known and indicate that locating the needed information may be difficult.

items (features in the present context, disks in RAID) produces a holistic result which is much better than the original components.

Finally, systems based on hypertext technology have been recently proposed. OncoDoc [267], a system for the assisted selection of guidelines for cancer treatment, encodes domain knowledge as a decision tree implemented through pages linked by hypertext links: the physician is presented with a sequence of choices which lead him to the guideline to be applied. These systems are less system-centric but user interaction is quite rigid and follows predefined paths. Creation and maintenance are expensive, and the addition of a single new pathology may well disrupt the entire structure, and consequently user familiarity with the system.

9.4.2 *Diagnosis Through Dynamic Taxonomies*

The visual exploratory search paradigm of dynamic taxonomy can be used for medical diagnosis [243, 253, 254] by allowing users to focus on a clinical sign, and see immediately all the other clinical signs related to the currently selected sign(s), in addition to the pathologies which exhibit that sign. The summary of clinical signs related to the current focus guides the user to refine his search while leaving him completely in charge of the interaction: it is his responsibility to select the appropriate signs among the available ones, though the system guides him by discarding unrelated signs which would result in dead-ends. Differently from traditional diagnostic systems, a dynamic taxonomy system is a transparent tool that assists the practitioner in an easily understood and unobtrusive way. The basic dynamic taxonomy model is extended here by taking into account the frequency of clinical signs in each disease:¹⁷ both pathologies and related clinical signs are ranked in order to provide a better guidance in the assessment of candidate pathologies.

Diagnosis by dynamic taxonomies is not targeted to specialists, but rather to practitioners not familiar with a specific area. We believe that the practical importance of a diagnostic system is in dealing with the unknown and unexpected: a physician will hardly use a computer system to diagnose a flue; diagnosing a rare tropical disease which he has never seen nor read about is a completely different story. Because of the very fast convergence to small results sets,¹⁸ the diagnostic activity is expected to be very quick and effective. In addition, by showing at each step all related signs, the physician can be confident to have considered all the relevant aspects in arriving at a diagnosis and, in fact, this approach seems the only way to guarantee a high quality standard on diagnosis.

¹⁷This technique actually extends the basic dynamic taxonomy model by using fuzzy membership [329] (see Sect. 5.6).

¹⁸See Sect. 3.2.

9.4.3 Application of Dynamic Taxonomies to the Diagnosis of Rare Diseases

It is especially interesting to apply these techniques to the diagnosis of rare diseases. Rare diseases affect an extremely low percentage of patients: less than 5 cases over 10,000 persons, but some rare diseases have less than ten known cases over the entire world population. The early screening of rare diseases is extremely important but has to be performed by physicians who are perforce not familiar with these pathologies. According to the first Italian *Report on rare diseases* [68], rare diseases are diagnosed mostly by specialists (80% of the cases), whereas pediatricians and family doctors diagnose 16.7% and 4.2% of the cases, respectively. Diagnosis by a specialist seems to be the key, but the failures in early screening make locating the right specialist quite difficult, so that 25% of the patients are diagnosed after three years and some patients after seven years.

A web-based dynamic taxonomies diagnostic system which assists physicians in diagnosing a rare disease from observed clinical signs is freely available at www.erare.di.unito.it. Over 2000 diseases, classified by clinical signs, are managed by the system, which works as a diagnostic front-end to the Orphanet database [201], currently the most authoritative database in this field.

The target is the general physician who has the responsibility of the early screening of rare diseases. The diagnostic problem is seen in terms of guided information access, rather than in terms of logic manipulation or traditional information retrieval. In the dynamic taxonomy used here, objects are pathological situations. Each object is classified under one or more concepts which represent features that characterize it. Such features may be clinical signs or causes of the pathology, but also locations in which a pathological situation is common, or age groups which exhibit it, etc. In short, all the available information which can be used to characterize an object can be used as features. Here, in addition to clinical signs, the *Age of Onset* is used. Signs and features are taxonomically organized in order to support abstraction (generalization/specialization) in accessing the knowledge base and to simplify the access to a specific feature, especially when many features are present.

For each disease, symptoms are classified by frequency on a scale of three values: Very frequent, Frequent and Occasional, following the Orphanet classification. These weights are used to rank candidate pathologies by decreasing average frequency. A graphical indication of the frequency of the clinical signs in the current focus is also displayed for each retrieved disease. In addition, frequency information is used to discriminate among signs. For each sign, a shade of red gives a visual indication of its average frequency in the current set of candidate pathologies. Since signs are hierarchically organized, each entry has a color indication of the average frequency of all its descendant signs.

The diagnostic process is an interactive exploration of the knowledge base. From the initial taxonomy (Fig. 9.8), the user will focus on the first symptom, e.g., *ataxia*. The zoom operation will produce a reduced taxonomy in which all and only the features related to ataxia are retained (see Fig. 9.9C). Related features are automatically computed from the extensional inference rule: thus *ataxia* and *cardiac*

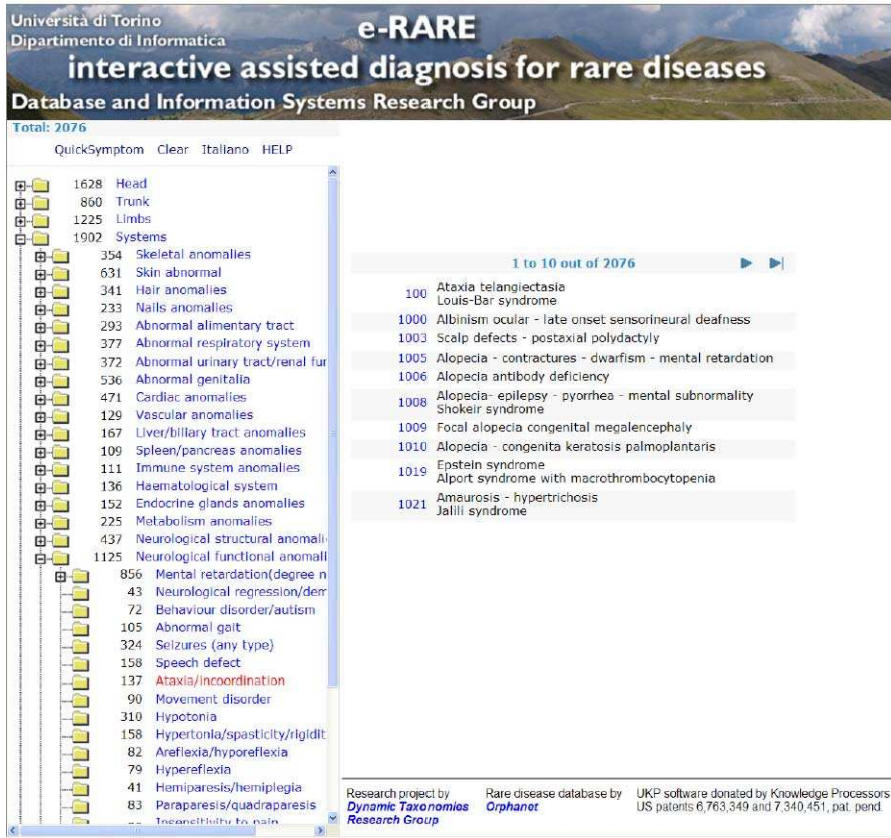


Fig. 9.8 Initial symptom taxonomy

anomalies are related because there is at least one pathology (14 pathologies in this case) in the knowledge base which is classified under both concepts (or one of their descendants). The 137 pathologies which exhibit *ataxia* are presented by decreasing frequency. In addition, shades of red are used to orient the user to high-probability signs: e.g., *muscle anomalies* are immediately perceived as more frequent than *cardiac anomalies* for *ataxia*.

The user of a traditional taxonomy would have to inspect the 137 diseases which exhibit *ataxia*. Here, instead, any symptom in the reduced taxonomy can be selected to restrict the focus. *Hyperglycemia* is now selected, and Fig. 9.10C shows the reduced taxonomy and the pathologies which exhibit both *ataxia* and *hyperglycemia*. The selection of two signs reduced the total number of candidate pathologies from 2,076 diseases to just 7 candidates.

The reduced taxonomy guides the user to reach his goal because it indicates related features and symptoms and focuses his attention on a complete list of features which are useful for diagnosis and must be considered. At the same time, the user can see unexpected features which can be used to discriminate among items.

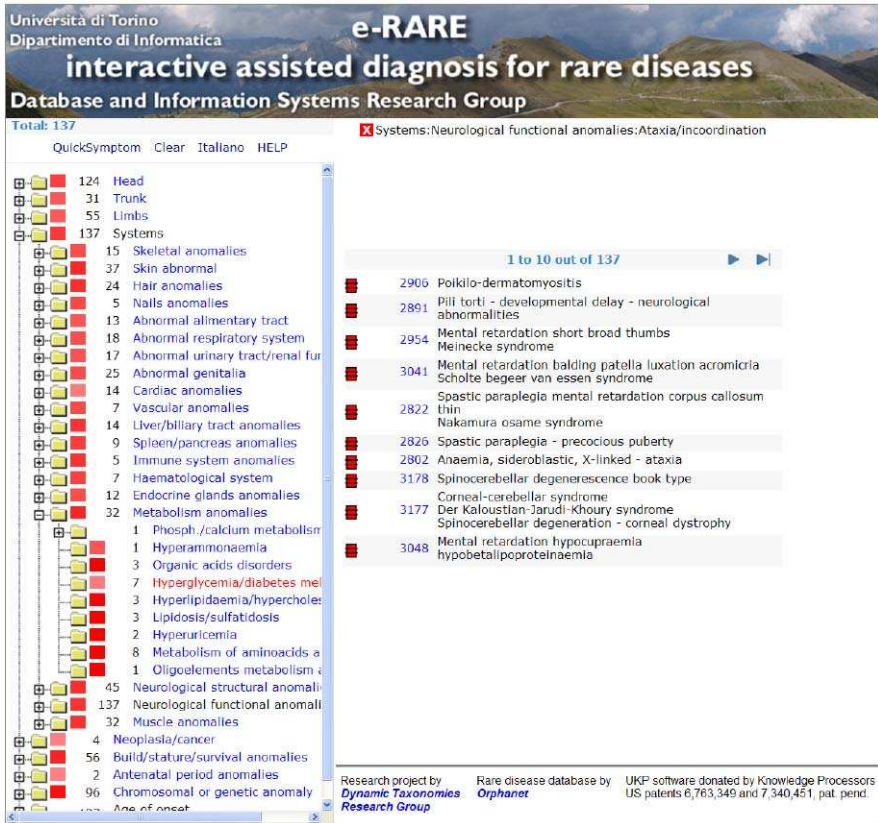


Fig. 9.9 Reduced taxonomy and diseases after a zoom on ataxia

Differently from traditional diagnosis, the diagnostic process can terminate at any time, and usually when the number of candidate items in the current focus is sufficiently small for manual inspection. The goal here is not to find the exact set of pathologies, but rather a sufficiently small set of candidate pathologies which the user will finally and independently evaluate.

Since the system was not available to the general public at the time of writing, its validation is preliminary because it was shown to and used by only a limited number of physicians. The initial feedback was positive: the summary by taxonomic signs was considered very important especially by physicians not familiar with rare diseases, who fully understood its importance in guiding the search, in the actual discovery of unknown candidate pathologies, and finally in the confidence of having considered all the aspects of the problem before arriving at a conclusion. This last benefit of dynamic taxonomy diagnostic systems is especially relevant, because diagnosis is usually hypothesis-driven so that it is very hard in practice to be exhaus-



Fig. 9.10 Reduced taxonomy and diseases after zooming on ataxia and hyperglycemia

tive and to guarantee a high quality. The interaction was considered very natural, and a physician commented that the system closely mimics the actual way a diagnosis is normally arrived at.

The relevance of this approach extends to diseases which can be endemic in some regions and rare elsewhere, but also to diseases which cannot be satisfactorily diagnosed, usually because some clinical signs are not being considered. A thorough and high-quality classification of pathologies by clinical signs and other features is obviously a *condicio sine qua non* for dynamic taxonomy diagnosis. The effort required is not trivial, but it is lower than in other approaches because each disease can be characterized independently, so that available collaborative techniques can be used. In addition, the results and the benefits of dynamic taxonomy diagnosis can be easily made available all over the world by the multilingual support offered by dynamic taxonomies.

Diagnosis through dynamic taxonomies is obviously applicable to different domains, and, most notably, to the diagnosis of mechanical, electronic and software malfunctions. The quick guided interactive thinning of dynamic taxonomies, together with easy non-disruptive updates of symptoms and malfunctions, makes this type of diagnosis attractive for online or phone assistance.

9.5 Digital Libraries and News Systems

Digital libraries and news systems are an obvious application area for dynamic taxonomies, especially if documents are already classified under a taxonomy. The first project in this area was the 1999 Final CS Encyclopedia¹⁹ prototype for the ACM Digital Library (DL). This project relied on the ACM Computing Classification System [13] which is used by authors to classify most of the papers in the ACM DL,²⁰ in addition to facets such as Author, Venue, Year, etc. The project provided superior access capabilities with respect to the DL access facilities available then and for several years afterwards.²¹

The Final CS Encyclopedia project was explicitly targeted to the ACM Digital Library because its taxonomic classification could provide an extremely interesting way of systematically tracking research trends and of encouraging interdisciplinary awareness in Computer Science, which is becoming more and more rigidly subdivided into different, independent areas, with the consequence of re-inventing, rather than reusing, the same solutions in different contexts. So the central information was really the topic classification. It was felt that the other facets were interesting (especially the Year facet, for the analysis of trends) but definitely second-order with respect to the “Topic” facet. Other document collections, such DBLP [169], CiteSeer [67] and IEEE Computer Society Digital Library [3], lacked a topic classification and were not considered acceptable candidates for our goals.

One of the unusual features of this project was that, in addition to a conventional web-server architecture, one of the prototypes used a distributed architecture which managed over 1,000 papers in a single Java jar requiring about 300 KB and containing indices, required data (titles, authors, links to the actual papers, etc.) and the dynamic taxonomy engine itself. This was an important architectural option at the time, because it offloaded the web server of the real-time computation of zooms which could be performed at the client side, while requiring a small bandwidth for the delivery (and continuing upgrade) of the information base. We are still convinced that a centralized architecture is suboptimal for applications with frequent access to large infobases (as in the case of the ACM DL), while a distributed architecture allows for a much cheaper infrastructure because it decreases both central computational requirements and network traffic.

Other applications of dynamic taxonomies to digital libraries are considerably later. In 2007, the DBLP repository was one of the targets of dynamic taxonomies/faceted search. Two applications are currently available on it, Complete-

¹⁹The name was inspired by the SF novel “The final encyclopedia” by Gordon R. Dickson.

²⁰ACM papers are usually classified under several topics.

²¹An implementation was offered to ACM in 2000 [257] as a contribution to the research community. The DL group was not interested, and dynamic taxonomy access was adopted eight years later. At that time, ACM’s CEO John R. White wrote [315]: “New search technology recently integrated has dramatically enhanced Digital Library searches and will enable a much richer, guided navigation experience of ACM’s (and other publishers’) content.” The late adoption of a technology that ‘dramatically enhances searches’ by one of the leading scientific and professional computer associations is a good example of the early acceptance problems encountered by dynamic taxonomies.

Search DBLP by the Max Planck Institut für Informatik [31] and FacetedDBLP by the L3S Research Center, University of Hannover [88]. The two systems differ in two respects. First, FacetedDBLP is non self-sufficient, according to our definition of SEAS in Sect. 3.1. The only initial access is through a text-retrieval query, probably because of performance reasons. CompleteSearch DBLP shows instead the most frequent values for each facet, and is therefore self-sufficient because SAES selectors can be used from the very beginning of the interaction.

Second, topics are totally absent in CompleteSearch DBLP, whereas an attempt to account for them is present in FacetDBLP, through the GrowBag [88] facet which suggests keywords from retrieved paper titles.

The new version of the ACM Digital Library available since November 2008 [315] is based on dynamic taxonomies and is essentially similar to FacetDBLP, though a different algorithm is used for suggesting additional query terms.

All these applications, though valuable in themselves, fall short of the goals of the Final CS Encyclopedia project discussed above. The heavy reliance of all three systems on information retrieval as a primary access paradigm, and the term suggestion mechanism as the only way to support semantic content have all the shortcomings which were discussed before, and certainly do not support knowledge-seeking activities in an effective way.

The central point in order to achieve the full power of dynamic taxonomies in this context is to allow the “topic” content of the infobase to be explicitly shown and manipulated by users through a specific facet. The techniques described in Chap. 7 can be used to create such a facet and to classify documents.

With a digital library such as ACM’s, classified by topics and with most citation data electronically available, one can efficiently support all three types of explorative tasks we discussed, since all of them are actually of interest for researchers and practitioners alike.

The *object-seeking* task usually occurs when writing a paper and needing to discuss previous research. A topic classification is invaluable in this context, because it eliminates the need for information retrieval and all the problems associated with it.

The *knowledge-seeking* task occurs when the user has to become familiar with a new area: he will be looking for authoritative papers for an initial picture of the new field. Here, the topic organization can be used to set the right context, a simple facet with citation counts gives an indication of which papers are authoritative, and the Year facet can be used to select initial or more current papers, as appropriate. Another important knowledge-seeking task is to find interdisciplinary links among material. With non-ambiguous terms, this can be done through information retrieval. As an example, a query on “hashing” will retrieve topics (and papers) in database systems, data structures, operating systems, etc.

Additional flexibility can be achieved by appropriate faceted design of the taxonomy. As an example, consider a facet taxonomy by topic (e.g., Database, Operating Systems) and a simple facet taxonomy by task (e.g., searching, sorting, caching). In this way, the fact that “caching” is a cross-disciplinary topic becomes immediately evident, since after zooming on it one will find Operating Systems>Virtual Memory, Database systems>Buffer Manager, etc.

Finally, the *wisdom-seeking* task means, in this context, the monitoring of topic trends and of group leaderships. What research areas are currently “hot”? What are the leading research centers in a specific area? Topic taxonomies are again fundamental in this context and the data-mining techniques described in Sect. 5.1 provide an adequate framework.

News systems share most of the exploration requirements of digital libraries, and stress the importance of push interactions. Especially in this area, the massive information flow requires the powerful filtering capabilities offered by dynamic taxonomies, but also asynchronous operations, in which the user specifies his own interests and the system advises him when interesting items arrive.

The Dbworld Xtended project [2, 242] implemented an integrated semantic dissemination system based on dynamic taxonomies, in which dynamic taxonomies were used to explore relevant information. The user could express his own interests at a conceptual level through the same taxonomic interface, so that precise push strategies could be implemented. Dbworld Xtended was used to manage announcements coming from DBWorld [1], one of the best-known computer science research mailing lists, but it represents a framework which can be easily adapted to the dissemination needs of very diverse application areas such as e-government, to e-commerce, personalized news, etc.

9.6 E-government

Although it is not really recognized by most current e-government research, the critical enabling factor for e-government and e-democracy is knowledge [246]. There cannot be any real e-democracy or e-participation if the citizen has not complete and easily accessible data on the issues he is asked to vote on or on which he is asked to participate, or on the laws and regulations he is required to abide. Other technologies, such as those which allow secure polling or voting, are required but do not solve the main problem: how can a large mass of information be made easily and effectively available to citizens so that they can make informed decisions.

In fact, the knowledge which e-governments are asked to manage and make available to citizens is quite articulate and of different types:

- *laws and regulations*

This represents the normative part of governments and although the Latin maxim “*ignorantia legis non excusat*”²² is a basic tenet of judgment, the number and complexity of laws and regulations has become so large that “ignorance of law is unavoidable” would probably be more accurate. There are laws at the town level, at the district level, at the regional level, at the country level, at the federal

²²Ignorance of law does not excuse.

level, at the world level. Some of these conflict, some are superseded by others in certain cases and in certain locations, and lawmakers at any jurisdictional level are always busy and very productive. One of the greatest opportunities of e-government is to overcome this information overload and to supply timely and complete information to everybody. The electronic availability of information *per se* is a minor aspect of this problem. Rather, effective, timely and accurate ways of disseminating information are needed.

- *government services*
Government e-services are one of the most frequent and critical points of contact between public administrations and citizens. Currently, typical e-government portals provide 100–200 e-services, a number growing as more and more government agencies go online. In addition to common services such as id cards, permits, they represent the only practical way of providing incentives and support to specific classes of citizens. For this reason, discovery of e-services, rather than plain retrieval, is a critical functionality in e-government systems.
- *government local promotion*
Governments, especially local ones, are using the web to provide a number of services which are mainly informative and aim at improving the quality of life of citizens and at promoting the local community. These services include, among others, job placement services, tourist information (hotels, restaurants, etc.), yellow pages to promote local industries and activities, etc. and are currently managed through traditional retrieval methods which are not effective in this context, where exploration is required instead.

Dynamic taxonomies can provide a framework to effectively manage all these types of knowledge [255]. Laws and regulations can be effectively described and accessed through dynamic taxonomies [246]. The push capabilities of dynamic taxonomies can be used to keep citizens informed of new laws and regulations which are interesting in a continuous way. Sacco [246] proposes a simplified dynamic taxonomy in the form:

- *Sector*,
a facet describing sectors of activity (agriculture, chemistry, etc.)
- *Location*,
a facet describing the location(s) to which a specific document applies
- *Subject*,
a facet describing the subject(s) (e.g., persons or companies, public companies, etc.) to which a specific document applies
- *Document type*,
a facet describing the type of document (law, regulation, opportunity, etc.)
- *Issuer*,
a facet describing the issuer (town, country, etc.)
- *Time of validity*,
a facet describing the time frame in which the object described is valid

which can be used to explore complex law and regulation corpora. The same grid can be used to express interests so that new relevant documents can be sent to the user.

Dynamic taxonomies can be used to describe e-services and allow the discovery of services which apply to the specific citizen [249, 251]. In [251], the static taxonomic organization of the Italian e-government portal [10] is discussed, as well as a simple directory of services agreed upon by the EU states [9]. In alternative to these architectures, a dynamic taxonomy organized into 7 facets is proposed:

1. *Services*:
services which the government supplies, organized by major areas such as the ones indicated in [9], e.g., Car registration, Income taxes, etc. This facet can be ordered by the most common services first, in order to speed up access for the “average” user;
2. *Events of life*:
events in the life of a citizen, such as the ones used in [10], e.g., Having a child, Studying, etc.;
3. *Type of information*:
this facet describes what type of information is available, such as online guides, online services, etc.;
4. *Location of person*:
abroad, or where in the nation. This facet is used to coordinate special services offered by specific municipalities, or regions, etc.;
5. *Type of citizenship*:
whether the person is a citizen of this state, or a citizen of a different nation. Citizens from specific groups of nations (e.g., EU citizens) might have special provisions. This facet can also be used to describe restricted citizenship rights, such as minor age or inhibition from public offices;
6. *Person with special rights*:
although the contents this facet can be represented in the following Person Profile facet, it is used to attract the user attention to specific categories she/he might belong to. Concepts in this facet may include age, sex, disabilities, personal relationships (e.g., single parents, married couples), etc.;
7. *Person profile*:
this facet describes other features of the person which can be used to reduce the number of services that apply. For instance, a useful criterion is Age: a person younger than 18, for instance, cannot have a firearm permit or a driving license, and cannot retire. Conversely, a senior citizen will not be directly interested in primary education.

The following example shows how a senior citizen could quickly find all the services and other relevant information which apply. In Fig. 9.11, the initial taxonomy is shown. Figure 9.12 shows the result of a zoom on Senior Citizen, which reduces the overall number of object from 539 to 67. The user can further restrict the total number of candidate documents by zooming on one of the facet values in Fig. 9.12, but the important fact is that the senior citizen has immediately a complete summary of all the services available to him, and of which he might not have been aware.

SERVICES	LIFE EVENTS	TYPE
Income taxes (20)	Having a child (12)	offline service (75)
Job search services (3)	Studying (35)	online service (375)
Social security contributions (12)	Working (67)	guide (89)
Personal documents (7)	Transportation (43)	
Car registration (4)	Housing (30)	
Application for building permission (25)	Family (55)	
Declaration to the police (15)	Paying taxes (67)	
Public libraries (9)	Going abroad (12)	
Certificates (40)	Health (60)	
Enrollment in higher education / university (7)	Sport (255)	
Change of address (4)	Police (35)	
Health related services (60)	Leisure and culture (44)	
	Helping others (12)	
	Retiring (16)	

WHERE ARE YOU	CITIZENSHIP	SPECIAL RIGHTS	PROFILE
abroad (15)	Italian (500)	Women (35)	Sex (539)
Italy (524)	EU (480)	Senior citizens (67)	Age (539)
	extra-EU (58)	Handicapped (86)	Education (539)
		Relationships (43)	

Fig. 9.11 E-government portal with a dynamic taxonomy on seven facets

SERVICES	LIFE EVENTS	TYPE
Income taxes (2)	Having a child (0)	offline service (5)
Job search services (0)	Studying (0)	online service (62)
Social security contributions (10)	Working (2)	guide (19)
Personal documents (0)	Transportation (13)	
Car registration (40)	Housing (7)	
Application for building permission (0)	Family (5)	
Declaration to the police (15)	Paying taxes (0)	
Public libraries (9)	Going abroad (0)	
Certificates (0)	Health (20)	
Enrollment in higher education / university (0)	Sport (0)	
Change of address (0)	Police (35)	
Health related services (20)	Leisure and culture (14)	
	Helping others (0)	
	Retiring (16)	

WHERE ARE YOU	CITIZENSHIP	SPECIAL RIGHTS	PROFILE
abroad (0)	Italian (67)	Women (5)	Sex (67)
Italy (67)	EU (4)	Senior citizens (67)	Age (67)
	extra-EU (0)	Handicapped (16)	Education (67)
		Relationships (0)	

Fig. 9.12 Reduced taxonomy after a zoom on Senior Citizens

Another important point [253] is that dynamic taxonomies make the evolution (insertion, deletion, update) of services especially easy to manage, because of the dynamic evaluation of relationships among concepts. In dynamic taxonomy designs, the addition of a new service or of a new information object only requires its clas-

sification according to the dynamic taxonomy. In traditional static designs, it can require the manual restructuring of a significant part of the existing portal.

Finally, most of the information provided by government local promotion (job placement services, tourist information, yellow pages, etc.) inherently require an exploratory access [244], and they can be easily managed by dynamic taxonomies. Current e-government portals manage these information by ineffective traditional retrieval methods, and, frequently, data are just presented as plain lists.

9.7 File Systems

We present LISFS, a file-system implementation of logical information systems (LIS), where objects are files or parts of files, and dynamic taxonomies replace (monodimensional) trees of folders. This has the benefit to make DT features available right now to existing applications such as editors or compilers. This implementation can easily be extended and specialized through a plug-in mechanism. We also present some applications in the field of personal databases (e.g., music, images, emails), and demonstrate that building specialized interfaces for visualizing databases can be done easily through LISFS navigation. LISFS has been implemented by Yoann Padioleau during his PhD [202, 203].

9.7.1 Implementation

LISFS (LIS File System) implements the principles of a LIS at the system-level under the interface of a file system. Objects are files or parts of files, and foci are presented as folders. Such a system approach makes DT access to files a commonplace, regardless of their types and whatever application is used [101, 202]. Modern OSes consider a file system as a set of methods that allows to manage files and folders. Under Linux, VFS (Virtual File System) acts as an interface of which every method is virtual. A file system is just an implementation of those methods; this is the case of LISFS, which runs over the Linux kernel. LISFS has a plug-in mechanism. There are two kinds of plug-ins: *logics* are used to describe, query and classify objects; *transducers* compute intrinsic properties of files. With LISFS, paths are seen as formulas. LISFS embeds boolean querying: / reads and, | reads or and ! reads not. In LISFS, common Linux shell commands can be reinterpreted in the framework of dynamic taxonomies.

- The Linux command `cd` is used to change the focus. If a relative path is given, the current query (the `pwd`) is refined (zoom-in). If an absolute path is given, the current query is replaced (pivot). For instance, the command `cd /music/disco|rock` would select disco or rock music files, and the command `cd !bad` would further refine the selection to those music files that are not tagged as bad.

- The Linux command `ls` has two modes. In the extensional mode, it returns the extension of the current focus as a list of files. In the intentional mode (the default), it returns the most general *selective* concepts in the dynamic taxonomy, where *selective* means that only concepts whose count is strictly lower than the size of the extension are returned. So, only a flat subset of the dynamic taxonomy is returned. This is necessary because the result of `ls` commands must be a list of sub-folders according to the VFS interface. Entering such a sub-folder corresponds to a zoom-in, and allows to see further sub-folders.
- Extrinsic properties can be given to files by moving them (command `mv`) to sub-folders. For instance, the command `mv 12.mp3 good/` moves the music file `12.mp3` to the sub-folder `good`, which means it is classified as `good`, in addition to existing properties (e.g., artist, title). Properties can also be removed, like in the command `mv good/12.mp3 .`, where the file `12.mp3` is moved from the sub-folder `good` to the current focus, and hence removed from the folder `good`.

It should be emphasized that LISFS operates on two levels. At the first level, called inter-file, objects are files. At the second level, called intra-file, objects are parts of files [203]. By design, querying and navigation within LISFS must be compatible with an interactive use. Current experiments show that it is the case up to an order of 100,000 objects. While the stress is put on a fast answering to queries, indexing might take longer. However, only the initial indexing of large data is expensive. Once it is done, the index is stored persistently on disk. If the data changes, re-indexing is incremental and faster. To achieve speed, LISFS uses inverted indexes to manage its meta-data. The indexes are hash tables, which are accessed with the Berkeley DB library [197].

9.7.2 Applications

9.7.2.1 Managing a Music Database

Music files are easily managed within LISFS. As a matter of fact, music files generally come with meta-data; e.g., MP3 files embed the name of the artist, the title of the song and so forth. Such meta-data can be extracted by a transducer and used to describe a music file. Using a traditional UNIX shell as an interface, the `cd` command queries the database and the `ls` command lists the related concepts as sub-folders, as shown in the example below:

```
[1] cd /lfs/music/year:[1980..1990]
[2] cd !genre:Comedy
[3] cd time:<7min
[4] cd .ext
[5] playmp3 *
...
[6] cd /lfs/music/genre:Disco/
[7] ls
```

```

artist:BeeGees/ artist:DonnaSummer/
...
year:1976/ year:1977/
...

```

Command 1 selects music files from the 80s (using an interval logic for years), command 2 excludes comedy from the selected files (! denotes negation) and command 3 selects only short songs (using a time logic). From then, the user can switch to the extension of its query (command 4) and listen to it (command 5). Command 6 selects disco music and command 7 then asks for related concepts. It can be seen that they can be used as legitimate navigation links so as to refine the current query. Those who prefer to use a graphical interface, as provided by popular tools such as iTunes, may find the use of a command-line interface in the above example awkward. This is not really a problem: since LISFS is implemented as a real file system, every application can benefit from it. Therefore, a music manager a la iTunes comes for free: it is barely a specialized interface over LISFS. Such an interface has been written [204]. This approach has two advantages. First, the code of the interface is very simple, since the hard work of indexing, querying and answering relies entirely on LISFS. Second, the resulting application is more powerful than most popular tools. For instance, iTunes allows the user to browse a music library first by genre and then by artist but not the reverse; this is no problem with our application.

9.7.2.2 Managing a Bibliography Database

The intra-file mode of LISFS can be used to manage a bibliography database stored in a Bib \TeX file. In this case, the objects are the entries of the Bib \TeX file, and a transducer extracts properties such as author, title, etc. LISFS was successfully used to manage a 100,000 lines Bib \TeX file. Navigation inside a Bib \TeX file is close to data-mining. For example, the related concepts resulting from a query can easily be used to look for frequent co-authors of a given author. Moreover, after performing a query, the user is presented a partial view of the original file, showing only the relevant entries. This view can not only be further queried, but also edited. Changes are propagated to the original file in a consistent way, and the view is consequently updated.

9.7.2.3 Managing E-mails

E-mails can also be managed within LISFS. Experiments were achieved on repositories holding up to 160,000 messages. A transducer extracts author, subject, thread, date, etc., from a message and this information is used to index e-mails. By assigning extrinsic properties to the e-mails, the user can organize its messages into custom concepts. It should be noticed that, unlike most e-mail tools, e-mails can naturally belong to several concepts.

A frequent user task is to look for old e-mails. Full-text indexing is required for this task, but a native LISFS implementation would be inefficient, since there would

be too many related concepts available. To alleviate this problem, Glimpse [176] is used for full-text indexing of the messages, and the file system interacts with Glimpse to process some queries. Queries of the form `contains:foo` call Glimpse to select files that contain the text `foo`; however, rather than showing these files, a dynamic taxonomy over them is computed. This demonstrates the flexibility of dynamic taxonomies by allowing the integration of external querying tools, without breaking the querying/navigation combination.

9.7.2.4 Homedir

Apart from the few examples described above, LISFS is used to manage many other kinds of files such as source code of programs, man pages, LaTeX files, etc. The idea of managing a user's entire home directory, which is nothing more than a collection of numerous and heterogeneous files, sounds appealing. As we saw in previous examples, the user benefits from the different applications of LISFS on various kinds of files. Moreover, LISFS allows him to perform transversal queries on different kinds of data. For example, let us consider a user looking for files that contain the word `synchronization` but who does not remember exactly their types nor their places. With a home directory managed under LISFS, he will be able to use the query `contains:synchronization` to select every data that mention the word `synchronization`, regardless whether it is in an e-mail, a documentation, a program, etc. Then, the system itself will propose navigation links for the user to refine the query.

9.7.3 Related Works

LISFS differs from relational databases in several points. The main advantages of LISFS are a schema-less organization of objects and a seamless integration between querying and navigation. Since navigation links computed by LISFS are expressed in the same language as queries, there is no need for the user to learn a query language. Moreover, object descriptions in LISFS are not limited to flat attributes; the use of specialized logics (e.g. date logic, string logic, interval logic) supports rich descriptions. We believe that the low-level implementation as a file system is another important point for LISFS. Being standalone applications, traditional databases are hard to interface with. On the contrary, LISFS is totally integrated in the operating system by design, so that LISFS technology is made available to every single application.

Non-hierarchical file systems were already proposed, e.g. SFS [120], Nebula [47], HAC [123]. Each of them brings some interesting features like automatic assignment of intrinsic properties, folders seen as queries, powerful querying or the notion of view. Nevertheless, none of them supports all those features at once, nor does provide such a seamless integration of querying and navigation as LISFS does.

Finally, none of these systems supports these principles at the intra-file level, as LISFS does.

LISFS can be seen as an answer to the problem of personal document retrieval. This issue has recently been addressed by the computer industry in two different ways. On the one hand, tools like Google Desktop indexes documents in a database; on the other hand, file system approaches, such as Apple's Spotlight, store indexes as file system attributes. Both approaches provide a searching tools in the form of a user application. LISFS goes a further step in that not only the indexes are stored in the file system, but also the file system *is* the searching tool.

9.8 Geographical Information Systems

Layer organization is the prevailing model for handling information in Geographical Information Systems (GIS). This structure gathers geographical data under a common theme e.g., soils, roads, water [164], and has become a standard for handling data. However, this structure is rigid as it implies partitioning geographic information in predefined categories, and usually having the same description schema for all the elements of a layer (pixels for raster data or features for vectorial data). As a consequence, data belonging to several themes are often duplicated in corresponding layers. Furthermore, the layer model is not really designed to manage relations between objects. If current GIS tools enable to query and to work on data distributed in several layers, pre-processing or repeated operations are often necessary.

Logical Information Systems (LIS) were proposed to avoid the rigidity of hierarchical data systems, and to merge querying facilities (as in databases) and navigation facilities (as in hierarchical file systems), under the same principles as dynamic taxonomies. LIS use logic in a uniform way to describe data, query it, to navigate through it and to update it. The LIS model has been implemented as a file system (see Sect. 9.7); this permits to integrate this new methodology into existing applications.

In this section, we explore how GIS applications can gain in flexibility by using LIS for handling geographical data. A prototype, GEOLIS, combining a navigation interface with data stored in a LIS, has been implemented by Olivier Bedel during his PhD [34, 35]. We successively present the data model and querying language, the interface and implementation of GEOLIS, and experiments on a real data set.

9.8.1 Data Model and Querying Language

In GEOLIS, a geographical feature is represented by an object, and logics are used to describe and query objects. We first define the description and querying languages. Then we present the real dataset that has been used to test the model.

Each object represents a geographical feature and is classified into a set of properties derived from the semantic attributes and a spatial description of the feature (geometry and location). These properties are given a name and may be atomic or valued.

There are several domains of values depending on the type of properties. They can be simple (string, integer, float) or composite (coordinates). Each domain is defined as a specialized logic (see Sect. 8.5) having its own language of formulas. Here is an example of a possible description of the French city of Rennes:

```
name:"Rennes" AND point AND
population:260000 AND position:(48.08,-1.68) AND
description:"administrative center of Brittany".
```

In this description, point is an atomic property about the geometric shape of the feature, population is an integer valued property, name is a string valued property and position is a coordinate valued property (latitude, longitude).

Each specialized logic defines taxonomies of patterns over its domain of values, in order to improve the expressivity of querying, and the progressiveness of navigation. For instance a string logic enables to build patterns like contains "administrative center". An interval logic on integer values provides patterns like `population:in [100000,300000]` or `population:>=200000`. The empty pattern, like in the query `population:` is the most general pattern of a logic. For instance, we have `population:265000` \sqsubseteq `population:>=200000` \sqsubseteq `population:`, where \sqsubseteq denotes logical subsumption (Definition 5.1). A coordinate logic can be defined as the product of two interval logics on real values (one for latitudes, and one for longitudes). It enables to build patterns corresponding to axis-aligned rectangular areas. For instance, Rennes's position (48.08, -1.68) is subsumed by $([40, 50], [-10, 10])$. Of course, attributes, values and patterns can be freely combined with the three Boolean connectors AND, OR, and NOT.

As an illustration of the GEOLIS data model, we now introduce the data set we used to make our experiments. It deals with the distribution of several species of rodents in Sahelo-Sudanian Africa. It is composed of one table where rows identify rodents and columns give descriptive information about these animals. This infobase is quite large (more than 20,000 individuals, potentially described by 92 attributes), and is the result of the merging of several databases provided and maintained by the French Institute for Research and Development (IRD) since 1980.²³ It has been mainly designed to study the actual distribution of rodents, and the possible causes affecting this distribution. Therefore, the description of each rodent mainly contains information about biometry (size, weight, sex, age), phylogeny (family, genus, specy), localization (position where the animal was captured, habitat), and period of capture. This semantic diversity, the various domains of values available (string, integer, float, coordinates) and the simple geometry (point) of the features make this base an interesting application for GEOLIS.

²³Inventaire et caractérisation des espèces de rongeurs sahélo-soudaniens, <http://www.mali.ird.fr/activites/inventaire.htm>.

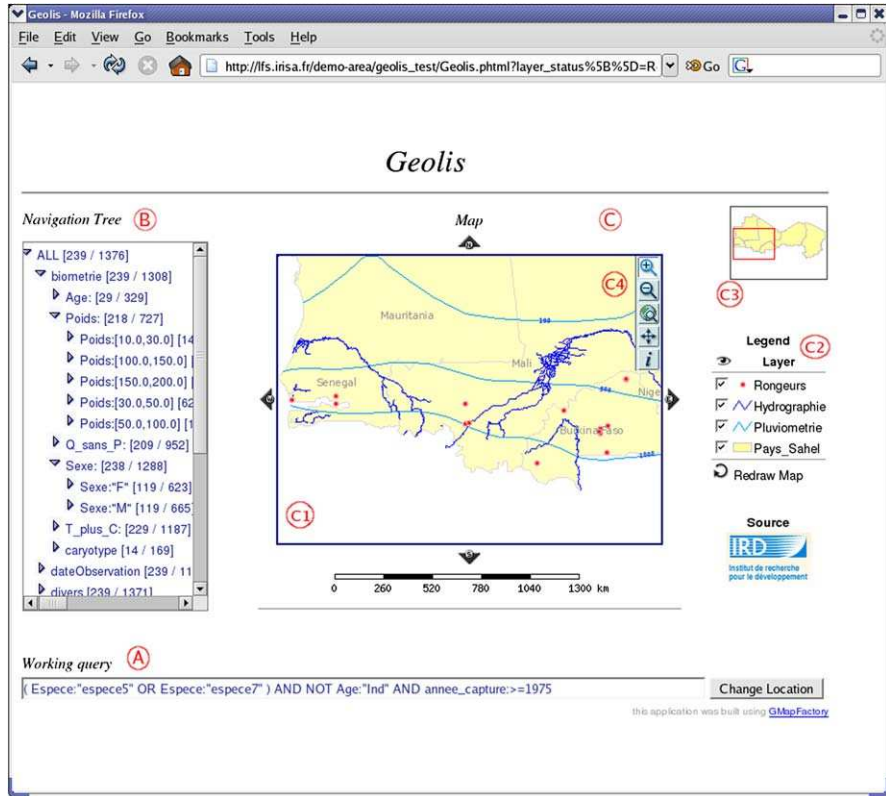


Fig. 9.13 The web interface of GEOLIS

9.8.2 Interface and Implementation of GEOLIS

The interface and interactions in GEOLIS are similar to what is presented in Chap. 4, except that the current selection of objects is presented on a map instead of as a list. Indeed, every object is a geographical feature with a position and a shape that can be rendered graphically. Figure 9.13C is a screenshot of the web interface of GEOLIS. The query box can be seen at the bottom (A), the dynamic taxonomy can be seen at the left (B), and the map area rendering the current selection of objects can be seen in the middle (C). The map area is a composed component. A main map (C1) including fixed background layers (administrative boundaries, hydrography and isohyetal lines) indicates by red points the position of rodents satisfying the current query. A legend (C2) details symbology of the main map and enables to specify which layers are visible on the main map. A keymap (C3) locates the boundaries of the main map on the Sahelian band. Last, standard map tools are available: pan, rectan-

gular zoom in, zoom out and zoom to full extents (C4). This first component comes almost unchanged from an existing interface.²⁴

In addition to the navigation modes presented in Sect. 4.3, GEOLIS defines a *geographical zoom*. It works like the traditional zoom-in and zoom-out, except it applies on a geographical region selected on the map. For instance, if a rectangular region is selected, it is translated into a coordinate pattern (e.g., `position: ([40, 50], [-10, 10])`), on which the zoom operation can be applied as usual. Its effect is to specialize or generalize the current query, and hence to restrict or extend the selection of objects. It must be distinguished from the map zoom (C4), whose effect is to show a smaller or larger part of the map, and has no effect on the current query and extension.

The GEOLIS prototype results from the coupling of several technologies from LISFS, web mapping and web domains. LISFS (see Sect. 9.7) constitutes the kernel of GEOLIS, where the geographical data to be explored is stored, i.e. the rodents base in our experimentations. The GEOLIS graphical interface is a web interface. The navigation tree and the working query box have been designed using the server side language PHP. The map area is built with the widely used map generator UMN MapServer. Between the several geographical formats supported by MapServer, we chose to use the Geographical Markup Language (GML) proposed by the Open-Geospatial Consortium. GML is an XML based format with public specifications. For our purpose, it has the advantages of gathering all information in one file whose XML based structure may be rearranged with regard to GML specifications. Furthermore, GML is supposed to become a standard for geographical data sharing. These technologies were not designed to work together. However, their combination in GEOLIS did not require any modification. Much of the work has been to interface them, i.e. to determine the geographical format the most appropriate for LISFS integration, writing the corresponding transducer, building logics devoted to geographical data and designing the navigation tree interface.

9.8.3 Experiments

The whole base (20,585 rodents with an average of 39 properties in each description) has been loaded in GEOLIS. Response times of navigation commands increased with the size of the context, but still allow human interaction (less than 10 seconds on an Intel Pentium M 2 GHz with 1 GB memory). The response mainly depends on LISFS, which is still under development and improvement.

Initial experiments in the rodents base highlight several occurrences of anomalous entries. These entries appear as properties with unexpected values and generally have a low support. In most cases, wrong entries do not belong to the domain of value of the properties, e.g. `sex: 49` instead of `sex: "F"` or `sex: "M"`. These anomalies result from errors in data collecting and merging, but sometimes indicates

²⁴MapServer, <http://www.mapserver.gis.umn.edu>.

uncertainty about the stored value e.g. `sex: "? "` or even `sex: "M? "`. Furthermore, GEOLIS also enables to identify consistency problems in the base, like properties having several semantically equivalent values, e.g. `sex: "M"` and `sex: "m"`.

Initially, spatial information in the rodents base was limited to the trapping position. So, to take into account the impact of other spatial factors on the distribution of rodents, some spatial relations, e.g. minimum distance from natural barriers (large rivers) or closest upper and lower isohyetal lines, have been processed for each rodents using external GIS tools. Then they have been translated into semantic properties. This enables to provide pseudo-spatial concepts in dynamic taxonomies. Furthermore, as the main map gives a visual representation of the location and concentration of rodents it could rapidly suggest relevant spatial query refinements during navigation.

As mentioned previously, the rodents base comes from an imperfect sampling. This has been observed in the navigation. For instance, the count of rodents for properties `pays` (country) and `annee_capture` (year of trapping) shows that half of information in the base comes from Senegal which clearly appears on the map to represent a small part of the studied area. The trapping frequency for a given country can be obtained by applying zoom-in on this country, and expanding years of trapping in the dynamic taxonomy. For example, we noticed that, in Chad, data have only been collected in year 2000, whereas in Mali and Senegal, data are available at least every two years.

Knowledge about data origin could enable to balance future results concerning rodents distribution. So we decided, as a first step, to study the sampling strategies in the database. For instance, we looked for connections between rodents trapped states (alive, dead), places and periods of capture. Within the navigation tree, we can restrict navigation to rodents trapped alive, and visualize, at the same time, properties `annee_capture` and `habitat`. For instance, selecting a particular habitat, e.g. `savane` (savanna), and looking at property `annee_capture` shows for each year, how many rodents were trapped alive in the savane. On the opposite, the distribution of trapping places concerning a particular year could be observed by selecting `annee_capture` and looking at `habitat`. This shows that GEOLIS is appropriate to quickly check distribution hypotheses implying several criteria. In only a few operations, we noticed that for rodents captured alive, the diversity of trapping places tightly depends on the period of capture. For instance, 85% of rodents found in savanna were trapped in year 2000, which also corresponds to more than 60% of rodents trapped that year. Low diversity in trapping places could be explained by trapping sessions led in order to confirm hypotheses implying location.

Chapter 10

Conclusions

Giovanni Maria Sacco and Yannis Tzitzikas

Frustra fit per plura quod potest fieri per pauciora
(It is futile to do with more things that which can be done with fewer)

Occam's razor

Dynamic taxonomies and faceted search support a new information access paradigm which allows the guided exploration of complex information bases, and bridges the gap between querying and browsing. Exploration through dynamic taxonomies is quickly understood by users. An appropriate faceted design produces compact and clear taxonomies which also afford a better correlation among concepts. Such correlations, based on the extensional inference rule, are used to accommodate both conceptual dynamicity and conceptual discovery, while keeping the underlying taxonomy relatively stable.

In this book, we provided the reader with a guide to all relevant aspects of dynamic taxonomies and faceted search. This is to be considered as an initial contribution to an area which we believe is rapidly evolving as new and ambitious applications emerge. Several research directions deserve further investigation:

- *New Applications and New Directions*

Although we have discussed several applications of dynamic taxonomies, we believe that the application scope of dynamic taxonomies is much wider. In particular, the application of dynamic taxonomies to knowledge and wisdom-seeking tasks is still in its infancy. We are especially interested in the potential of dynamic taxonomies to foster creativity, by revealing unexpected correlations among different concepts, and allowing to see and evaluate knowledge from different points of view.

Another important direction is preventing the rigid separation of knowledge into different, overspecialized sub-sub-disciplines. As we mentioned, this was the driving idea behind the 'Final CS Encyclopedia' project. The current status of our education and research has diverged from the ideal of the Renaissance man, whose education encompassed arts and sciences, to the point that the complexity of science seem to be such a centrifugal force that different disciplines often do not even share a common 'language'. A solid shared framework, and the automatically inferred connections among concepts can provide initial foundations for systems which can help to reverse this trend, and is especially relevant for e-learning.

- *Methodological Issues*

There are huge and continuously increasing amounts of digital information in various formats and systems. To aid the provision of faceted and dynamic taxonomy-based access services over existing sources, we need to develop methodologies, auxiliary tools, and best practices for various frequently occurring scenarios. For instance, the process of deriving zoom points to users (instead of requesting them to formulate complex queries in sophisticated query languages), has to be elaborated for various kinds of representation frameworks and query languages (although we have already described some cases in this book). Furthermore, the provision of integrated access to different sources raises issues analogous to those studied in information integration (e.g. extraction, cleaning).

- *Efficiency, Scalability, and Distributed Architectures*

Systems managing real-time interactions on a few million objects exists, and practical implementations issues and solutions are described in this book. However, as the application scope increases, the size of the information bases to be managed will increase as well. Real-time exploration of billions of objects is the next challenge.

Another open question is whether a distributed architecture is more efficient and scalable than the current, centralized approach. In addition, the application of dynamic taxonomies to large scale peer-to-peer environments seems to be a promising approach, since current peer-to-peer networks rely on rather primitive search facilities.

- *Taxonomy Design, Object Classification, and Quality ‘in-the-Large’*

Rapidly evolving multimillion information bases are challenging in many respects, especially when data are unstructured. Therefore, automatic classifiers and taxonomy designers/maintainers should be explored.

A manual, cooperative, and distributed approach to both taxonomy design and maintenance, and object classification seems also appropriate in many applications, and it is already successfully used in tagging systems. The problems arising in this context involve the concurrent evolution of the taxonomy, where coherence and non-redundancy are an obvious requirement, and the classification of objects, which involves different persons and perspectives on data and is especially difficult to maintain as the taxonomy evolves. Moreover, the quality/accuracy of the overall system must be guaranteed.

- *Visualization and Adaptability*

Several visualization strategies are discussed in this book, but this research area is in its initial phase, especially with respect to wisdom-seeking tasks, and for interactions on mobile devices with small displays (PDA's and cell phones). The compactness and fast convergence of dynamic taxonomies make them especially interesting in this context. Their easy adaptability by focus pre-setting can make them a good choice for ambient intelligence as well.

- *Social Aspects*

Current social networking systems rely on conventional techniques for ‘searching’ the network. As a consequence, the retrieval of potential contacts is usually difficult, time-consuming, and error-prone. The first results of a (yet unpublished) pilot study on a subset of LinkedIn (a social network for professionals,

at `linkedin.com`) shows significant benefits for the users of a dynamic taxonomy approach.

Social aspects also include extensions to the collaborative filtering and recommendation techniques discussed in this book.

- *Standardization*

As we noted, standardization on exchange formats, protocols, and services is almost nil, and a substantial effort on this topic is required. Standards which have emerged from other areas (e.g., Digital Libraries, Semantic Web) can be adopted and adapted to the needs of dynamic taxonomies.

Dynamic taxonomies represent an exercise in conceptual economy, both from the modeling and from the user interaction point of view, which are considered holistically. An important consequence of this approach is that users easily understand the model to the extent that it quickly becomes transparent, and the interaction natural and fully under user control. This makes dynamic taxonomies accepted in applications, such as medical diagnosis, where knowledge-based systems and agent-mediated interaction have not been successful. More generally, it shows that models and paradigms which do not account for the user and are too complex to be quickly understood by casual users are likely to be unsuccessful in practice. Conceptual economy also allows researchers to extend the model and to apply it to different problems. Although extensions perforce increase complexity, the minimal conceptual foundations of dynamic taxonomies allow significant extensions of the model without negative effects on the ease of interaction.

The application range of dynamic taxonomies is quite wide, and includes the three major exploration tasks we identified: pragmatic object selection (typical of e-commerce applications), knowledge-seeking tasks (typical of knowledge management), and emerging wisdom-seeking tasks, in which the inner laws of the information base are the focus. The rapid acceptance of this paradigm in e-commerce, where dynamic taxonomies are the *de facto* standard for product selection, shows the vast potential of this approach in other, still untapped areas. We believe that dynamic taxonomies and faceted search will become the paradigm of choice in all those ‘search’ applications which are in fact exploratory in nature, and in which user interaction is required.

“*EN OIΔA, OTI OYΔEN OIΔA*”

(*One thing only I know, and that is that I know nothing*)

Socrates, 469 BC–399 BC

References

1. Dbworld, <http://www.cs.wisc.edu/dbworld/browse.html>.
2. Dbworld Xtended, <http://www.dbworldx.di.unito.it>.
3. IEEE Computer Society CS Digital Library, <http://www2.computer.org/portal/web/csdl>.
4. Knowledge Processors, SF book demo, <http://www.knowledgeprocessors.com/frame.asp?ID=19>.
5. Monster, <http://www.monster.com>.
6. RDF: Resource Description Framework, <http://www.w3.org/RDF/>.
7. SIMILE: Longwell RDF Browser, <http://simile.mit.edu/wiki/Longwell>.
8. SPARQL Query Language for RDF, W3C Candidate Recommendation, 6 April 2006, <http://www.w3.org/TR/rdf-sparql-query/>.
9. The European Union, Basic public services agreed upon by EU member states, http://europa.eu.int/information_society/eeurope/2002/action_plan/pdf/basicpublicservices.pdf.
10. The Italian Government Portal, <http://www.italia.gov>.
11. XTM: Topic Maps, <http://topicmaps.org/>.
12. M. Abrol, N. Latache, U. Mahadevan, J. Mao, R. Mukherjee, P. Raghavan, M. Tourn, J. Wang, and G. Zang. Navigating large-scale semi-structured data in business portals. In *Proceedings of the 27th VLDB Conference*, 2001.
13. ACM. ACM Computing Classification System, <http://www.acm.org/about/class/>, 1998.
14. E. Adar, D. Karger, and L. Stein. Haystack: per-user information environments. In *Conference on Information and Knowledge Management*, 1999.
15. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, 2005.
16. E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior. In *SIGIR '06: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.
17. R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. *SIGMOD Records*, 18(2):253–262, 1989.
18. R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. ACM Press, New York, 1993.
19. J. F. Allen, C. I. Guinn, and E. Horvitz. Mixed-initiative interaction. *IEEE Intelligent Systems*, 14(5):14–23, 1999.
20. G. Amato and C. Meghini. Faceted content-based image retrieval. In G. M. Sacco, editor, *DEXA Workshop on Dynamic Taxonomies and Faceted Search (FIND'08)*. IEEE Computer Society, Los Alamitos, 2008.
21. A. Analyti and I. Pachoulakis. Logic programming representation of the compound term composition algebra. *Fundamenta Informaticae*, 73(3):321–360, 2006.
22. A. Andrejko, M. Barla, and M. Tvarožek. Comparing ontological concepts to evaluate similarity. In *Tools for Acquisition, Organisation, and Presenting of Information and Knowledge: Research Project Workshop*, September 2006.
23. G. Antoniou and F. Van Harmelen. *A Semantic Web Primer*. MIT Press, Cambridge, 2004.
24. G. Antoshenkov. Byte-aligned bitmap compression. Technical Report, Oracle Corp., 1994.

25. L. Ardissono, A. Goy, G. Petrone, and M. Segnan. Personalization in business-to-customer interaction. *Communications of the ACM*, 45(5):52–53, 2002.
26. P. Avesani and A. Agostini. A peer-to-peer advertising game. In *Proceedings of the 1st International Conference on Service Oriented Computing, ICSOC-03*, Trento, Italy, December 2003.
27. F. Baader. Engineering of logics for the content-based representation of information. In J. J. Alferes and J. A. Leite, editors, *Joint Eu. Conf. Logics in Artificial Intelligence*. LNCS, vol. 3229. Springer, Berlin, 2004.
28. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Reading, 1999.
29. H. Bast. CompleteSearch DBLP, <http://dblp.mpi-inf.mpg.de/dblp-mirror/index.php>.
30. H. Bast and I. Weber. When you're lost for words: Faceted search with autocompletion. In A. Broder and Y. Maarek, editors, *Workshop on Faceted Search (FIND'06) at SIGIR'06*, Seattle, USA, August 2006. ACM Press, New York, 2006.
31. H. Bast and I. Weber. The complete search engine: Interactive, efficient, and towards IR & DB integration. In *CIDR*, 2007.
32. M. J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407–424, 1989.
33. B. Beckert and J. Posegga. leanTAP: Lean, tableau-based deduction. *Journal of Automated Reasoning*, 11(1):43–81, 1995.
34. O. Bedel, O. Ridoux, and E. Quesseveur. Combining logical information system and OpenGIS tools for geographical data exploration. In *Int. Conf. on Free and Open Source Software for Geoinformatics*, September 2006.
35. O. Bedel, S. Ferré, O. Ridoux, and E. Quesseveur. GEOLIS: A logical information system for geographical data. *Revue Internationale de Géomatique*, 17(3–4):371–390, 2008.
36. R. M. Bell and Y. Koren. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.
37. O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. Shekita, B. Sznajder, and S. Yogev. Beyond basic faceted search. In *Proceedings of the International Conference on Web Search and Web Data Mining*. ACM Press, New York, 2008.
38. G. Berio, M. Harzallah, and G. M. Sacco. Portals for integrated competence management. In A. Tatnall, editor, *Encyclopaedia of Portal Technologies and Applications*. IGI Global, Hershey, 2007.
39. G. Berio, M. Harzallah, and G. M. Sacco. e-HRM—integrating architectures for competence management. In T. Torres-Coronas and M. Arias-Oliva, editors, *Encyclopedia of Human Resources Information Systems: Challenges in e-HRM*. IGI Global, Hershey, 2008.
40. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284:35–43, 2001.
41. K. Bharat. Searchpad: Explicit capture of search context to support web search. In *Proceedings of 9th International WWW Conference*, 2000.
42. W. Bintine and H. Tirri. Multi-faceted learning for web taxonomies. In *KD-NET 2002: 2nd Workshop on Semantic Web Mining*, 2002.
43. C. Bizer and R. Cyganiak. The TriG Syntax, <http://www4.wiwiss.fu-berlin.de/bizer/TriG/>.
44. D. C. Blair and M. E. Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the ACM*, 28(3):289–299, 1985.
45. R. Bodner and M. Chignell. Dynamic hypertext: querying and linking. *ACM Computing Surveys*, 31(4es):15, 1999.
46. S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering*. IEEE Computer Society, Washington, 2001.
47. C. M. Bowman, C. Dharap, M. Baruah, B. Camargo, and S. Potti. A file system for information management. In *Int. Conf. Intelligent Information Management Systems (ISMM)*, 1994.
48. P. Brézillon. Context in problem solving: a survey. *The Knowledge Engineering Review*, 14(1):1–34, 1999.

49. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Technical report, Microsoft Research, One Microsoft Way, Redmond, WA 98052, 1998.
50. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference (WWW 1998)*, 1998.
51. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*. ACM Press, New York, 1997.
52. M. Bruls, K. Huizing, and J. van Wijk. Squarified treemaps. In *Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG 2000)*. IEEE Press, New York, 2000.
53. P. Brusilovsky and C. Tasso. Preface to special issue on user modeling for web information retrieval. *User Modeling and User Adapted Interaction* 14(2–3):147–157, 2004.
54. B. G. Buchanan and E. H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Series in Artificial Intelligence. Addison-Wesley, Boston, 1984.
55. J. Callan. Document filtering with inference networks. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996.
56. D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proc. of the 2001 Int. Semantic Web Working Symposium (SWWS 2001)*, Stanford University, California, USA, July 30–August 1, 2001.
57. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer Academic, Dordrecht, 1998.
58. A. F. Cárdenas. Analysis and performance of inverted data base structures. *Communications of the ACM*, 18(5):253–263, 1975.
59. C. Carpineto and G. Romano. Exploiting the potential of concept lattices for information retrieval with CREDO. *Journal of Universal Computation*, 10(8):985–1013, 2004.
60. P. Cellier, S. Ferré, O. Ridoux, and M. Ducassé. A parameterized algorithm to explore formal contexts with a taxonomy. *International Journal of Foundations of Computer Science*, 19(2):319–343, 2008.
61. K. Chakrabarti, S. Chaudhuri, and S.-W. Hwang. Automatic categorization of query results. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM Press, New York, 2004.
62. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, 1997.
63. H. Chen and D. R. Karger. Less is more: Probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*. ACM Press, New York, 2006.
64. P. Chen. The entity–relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
65. H. Chernoff and E. L. Lehmann. The use of maximum likelihood estimates in χ^2 tests for goodness of fit. *The Annals of Mathematical Statistics*, 35(3):579–586, 1954.
66. J. Chomicki. Preference formulas in relational queries. *ACM Transactions on Database Systems*, 28(4):427–466, 2003.
67. CiteSeer. Scientific literature digital library, <http://citeseerx.ist.psu.edu/>.
68. Cittadinanzattiva. Report sulle malattie rare, <http://www.cittadinanzattiva.it>, 2008.
69. E. F. Codd. *The Relational Model for Database Management: Version 2*. Addison-Wesley, Boston, 1990.
70. E. F. Codd, S. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to user-analysts: An IT Mandate, 1993.
71. W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML 1995)*, 1995.

72. R. J. Cole, P. W. Eklund, and G. Stumme. Document retrieval for email search and discovery using formal concept analysis. *Applied Artificial Intelligence*, 17(3):257–280, 2003.
73. L. Console, D. T. Dupre, and P. Torasso. A theory of diagnosis for incomplete causal models. In *Proc. 11th IJCAI*, 1989.
74. W. B. Croft. Relevance feedback and personalization: A language modeling perspective. In *Proceedings of Second DELOS Workshop: Personalization and Recommender Systems in Digital Libraries*, 2001.
75. W. B. Croft and J. Lafferty. *Language Modeling for Information Retrieval*. Kluwer Academic, Norwell, 2003.
76. D. R. Cutting, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the 16th Annual International Conference on Research and Development in Information Retrieval (SIGIR'93)*. ACM Press, New York, 1993.
77. D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*. ACM Press, New York, 1992.
78. R. Dachselt, M. Frisch, and M. Weiland. Facetzoom: a continuous multi-scale widget for navigating hierarchical metadata. In *CHI '08: Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, New York, 2008.
79. W. Dakka, P. Ipeirotis, and K. R. Wood. Automatic construction of multifaceted browsing interfaces. In *Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM 2005)*, 2005.
80. M. Dalal. Investigations into a theory of knowledge base revision. In *Conference on Artificial Intelligence, AAAI-88*, St. Paul, Minesota, August 1988.
81. R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval: approaches and trends of the new age. In *MIR '05: Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*. ACM Press, New York, 2005.
82. J. de Kleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2–3):197–222, 1992.
83. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology*, 41(6):391–407, 1990.
84. L. F. del Cerro, D. Fauthoux, O. Gasquet, A. Herzig, D. Longin, and F. Massacci. Lotrec: The generic tableau prover for modal and description logics. In R. Goré, A. Leitsch and T. Nipkow, editors, *IJCAR. LNAI*, vol. 2083. Springer, Berlin, 2001.
85. J. Delgado and N. Ishii. Memory-based weighted majority prediction for recommender systems. In *ACM SIGIR'99 Workshop on Recommender Systems*, 1999.
86. B. Demo and A. Angius. E-learning: Coupling course management systems and dynamic taxonomies. In *DEXA Workshops*, 2007.
87. M. Dewey. *Dewey Decimal Classification and Relative Index*, 21st edition. OCLC, Dublin, 1997.
88. J. Diederich, W.-T. Balke, and U. Thaden. Demonstrating the semantic growbag: automatically creating topic facets for facettedbip. In *JCDL*, 2007.
89. X. Dong and A. Y. Halevy. A platform for personal information management and integration. In *Conference on Innovative Data Systems Research*, 2005.
90. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.
91. M. Dörk, S. Carpendale, C. Collins, and C. Williamson. Visgets: Coordinated visualizations for web-based information exploration and discovery. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1205–1212, 2008.
92. J. Ducrou, B. Vormbrock, and P. W. Eklund. FCA-based browsing and searching of a collection of images. In *Int. Conf. Conceptual Structures. LNCS*, vol. 4068. Springer, Berlin, 2006.

93. S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM Conference on Information and Knowledge Management (CIKM 1998)*, 1998.
94. S. Dumais, E. Cutrell, R. Sarin, and E. Horvitz. Implicit queries (IQ) for contextualized search. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)*. ACM Press, New York, 2004.
95. E. B. Duncan. A faceted approach to hypertext. In R. McAleese, editor, *HYPERTEXT: Theory into Practice*. Intellect Books, Exeter, 1989.
96. T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.
97. R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*, 2nd edn. Benjamin/Cummings, Redwood City, 1994.
98. Endeca. <http://www.endeca.com>.
99. M. A. Hearst et al. The Flamenco Search Interface Project, <http://bailando.sims.berkeley.edu/flamenco.html>.
100. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, 1998.
101. S. Ferré and O. Ridoux. A file system based on concept analysis. In Y. Sagiv, editor, *Int. Conf. Rules and Objects in Databases*. LNCS, vol. 1861. Springer, Berlin, 2000.
102. S. Ferré and O. Ridoux. A logical generalization of formal concept analysis. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*. LNCS, vol. 1867. Springer, Berlin, 2000.
103. S. Ferré and O. Ridoux. Searching for objects and properties with logical concept analysis. In H. S. Delugach and G. Stumme, editors, *Int. Conf. Conceptual Structures*. LNCS, vol. 2120. Springer, Berlin, 2001.
104. S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 40(3):383–419, 2004.
105. S. Ferré and O. Ridoux. Logic functors: A toolbox of components for building customized and embeddable logics. Research report RR-5871, INRIA, March 2006 (103 p.).
106. S. Ferré and O. Ridoux. Logical information systems: from taxonomies to logics. In *Int. Work. Dynamic Taxonomies and Faceted Search (FIND)*. IEEE Computer Society, Los Alamitos, 2007.
107. E. Filatova and V. Hatzivassiloglou. Marking atomic events in sets of related texts. In *Recent Advances in Natural Language Processing III (RANLP 2003)*, 2003.
108. L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppim. Placing search in context: The concept revisited. In *Proceedings of 10th International World Wide Web*, 2001.
109. P. C. Fishburn. *Utility Theory for Decision Making*. Wiley, New York, 1970.
110. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *Computer*, 28(9):23–32, 1995.
111. Flickr. <http://www.flickr.com>.
112. G. Flouris. On belief change and ontology evolution. PhD thesis, Computer Science Department, University of Crete, Greece, 2006.
113. S. Funk. Try this at home. <http://sifter.org/~simon/journal/20061211.html>, December 2006.
114. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Heidelberg, 1999.
115. P. Gärdenfors. Belief revision: an introduction. In P. Gärdenfors, editor, *Belief Revision*, pages 1–20. Cambridge University Press, Cambridge, 1992.
116. P. Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, 2000.
117. J. Gemmell, G. Bell, and R. Lueder. Mylifebits: a personal database for everything. *Communications of the ACM*, 49(1):88–95, 2006.
118. K. Georgila, J. Henderson, and O. Lemon. User simulation for spoken dialogue systems: Learning and evaluation. In *Proceedings of the 9th International Conference on Spoken Language Processing (INTERSPEECH-ICSLP 2006)*, 2006.

119. T. Gevers and A. W. M. Smeulders. A comparative study of several color models for color image invariant retrieval. In *Proceedings of the First International Workshop on Image Databases and Multi-Media Search (IDB-MMS '96)*, 1996.
120. D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O. Toole Jr. Semantic file systems. In *ACM Symp. Operating Systems Principles (SOSP)*. ACM Press, New York, 1991.
121. R. Godin, R. Missaoui, and A. April. Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. *International Journal of Man-Machine Studies*, 38(5):747–767, 1993.
122. Google Scholar Google. <http://scholar.google.com/>.
123. B. Gopal and U. Manber. Integrating content-based access mechanisms with hierarchical file systems. In *Proceedings of the Third Symposium Operating Systems Design and Implementation*. USENIX Association, Berkeley, 1999.
124. K. Groenbaek and R. H. Trigg (eds.). Hypermedia. *Communications of the ACM*, 37(2), 1994.
125. Classification Research Group. The need for a faceted classification as the basis of all methods of information retrieval. In L. M. Chan et al., editors, *Theory of Subject Analysis: A Source Book*. Libraries Unlimited, Littleton, 1985.
126. T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic, Dordrecht, 1994.
127. M. J. Halvey and M. T. Keane. An assessment of tag presentation techniques. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*. ACM Press, New York, 2007.
128. S. O. Hansson. What is Ceteris Paribus preference? *Journal of Philosophical Logic*, 25(3):307–332, 1996.
129. V. Hatzivassiloglou, L. Gravano, and A. Maganti. An investigation of linguistic features and clustering algorithms for topical document clustering. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000)*, 2000.
130. M. A. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49(4):59–61, 2006.
131. M. A. Hearst. Design recommendations for hierarchical faceted search interfaces. In *Proc. SIGIR 2006, Workshop on Faceted Search*, August 2006.
132. M. A. Hearst. Oral presentation at the faceted search workshop. In *ACM SIGIR 2007*, 2007.
133. M. Hearst. UIs for faceted navigation: Recent advances and remaining open problems. In *Workshop on Computer Interaction and Information Retrieval, HCIR*, Redmond, WA, October 2008.
134. M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Yee. Finding the flow in web site search. *Communications of the ACM*, 45(9):42–49, 2002.
135. J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, 2007.
136. J. Heer, M. Agrawala, and W. Willett. Generalized selection via interactive query relaxation. In *Computer-Human Interaction (CHI)*. ACM Press, New York, 2008.
137. W. P. Heising. Note on random addressing techniques. *IBM Systems Journal*, 2(2):111–116, 1963.
138. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, New York, 1999.
139. M. Hildebrand, J. van Ossenbruggen, and L. Hardman. /facet: A browser for heterogeneous semantic web repositories. In *International Semantic Web Conference*, 2006.
140. H. Hirsh, C. Basu, and B. D. Davison. Learning to personalize. *Communications of the ACM*, 43(8):102–106, 2000.

141. T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Francisco, 1999.
142. M. Holí and E. Hyvönen. Fuzzy view-based semantic search. In *ASWC*, 2006.
143. I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In B. Nebel, editor, *Int. Joint Conf. Artificial Intelligence*. Morgan Kaufmann, San Mateo, 2001.
144. D. Huynh and D. Karger. Parallax and companion: Set-based browsing for the data web, submitted to WWW 2009.
145. D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*. ACM Press, New York, 2007.
146. E. Hyvönen, K. Viljanen, and O. Suominen. HealthFinland—Finnish health information on the semantic web. In *ISWC/ASWC*, 2007.
147. E. Hyvönen, E. Mäkelä, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. MUSEUMFINLAND—Finnish Museums on the Semantic Web. *Journal of Web Semantics*, 3(2–3):224–241, 2005.
148. Google Inc., <http://www.google.com>.
149. International Organization for Standardization. Documentation—Guidelines for the establishment and development of monolingual thesauri, ISO 2788-1986, 1986.
150. P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.
151. H. V. Jagadish. A retrieval technique for similar shapes. In *SIGMOD '91: Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*. ACM Press, New York, 1991.
152. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
153. G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, 2003.
154. R. Jin, J. Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, New York, 2004.
155. D. Karger and M. Schraefel. The pathetic fallacy of RDF, http://swui06/papers/karger/pathetic_fallacy.html, 2006.
156. A. K. Karlson, G. G. Robertson, D. C. Robbins, M. P. Czerwinski, and G. R. Smith. Fathumb: a facet-based interface for mobile search. In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, New York, 2006.
157. G. Karvounarakis, V. Christophides, and D. Plexousakis. RQL: A declarative query language for RDF. In *Eleventh International World Wide Web Conference (WWW)*, Hawaii, USA, 2002.
158. H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings KR-91*, 1991.
159. G. Kobilarov and I. Dickinson. Humboldt: exploring linked data. In *Linked Data on the Web Workshop (LDOW2008) at WWW2008*, Beijing, China, 2008.
160. J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
161. S. Kopidaki, P. Papadakos, and Y. Tzitzikas. STC+ and NM-STC: Two novel online results clustering methods for web searching. In *Proceedings of the 10th International Conference on Web Information Systems Engineering, WISE'09*, Poznan, Poland, October 2009.
162. J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *Proceedings of the 17th International Conference on the World Wide Web (WWW '08)*, 2008.
163. B. Kules. Supporting exploratory web search with meaningful and stable categorized overviews. PhD thesis, University of Maryland at College Park, 2006.
164. R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Academic Press, New York, 1992.

165. M. Lenzerini. Data integration: a theoretical perspective. In *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002)*, Madison, Wisconsin, USA, 2002.
166. E. Levin, R. Pieraccini, and W. Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, 2000.
167. D. Lewis. Applying support vector machines to the TREC-2001 batch filtering and routing tasks. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, 2002.
168. D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1996)*, 1996.
169. M. Ley. DBLP computer science bibliography, <http://www.informatik.uni-trier.de/ley/db/index.html>.
170. R. Li, S. Bao, Y. Yu, B. Fei, and Z. Su. Towards effective browsing of large scale social annotations. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*. ACM Press, New York, 2007.
171. C. Lindig. Concept-based component retrieval. In *IJCAI95 Workshop on Formal Approaches to the Reuse of Plans, Proofs, and Programs*, 1995.
172. P. H. Lindsay and D. A. Norman. *Human Information Processing*. Academic Press, New York, 1977.
173. J. Lloyd. *Foundations of Logic Programming (Symbolic Computation/Artificial Intelligence)*. Springer, Berlin, 1987.
174. D. MacQueen. An implementation of Standard ML modules. In *LISP and Functional Programming*, 1988.
175. E. Mäkelä, E. Hyvönen, and S. Saarela. Ontogator—a semantic view-based search engine service for web applications. In *International Semantic Web Conference*. LNCS, vol. 4273. Springer, Berlin, 2006.
176. U. Manber and S. Wu. GLIMPSE: A tool to search through entire file systems. In *USENIX Winter Conf. USENIX*, 1994.
177. B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.
178. C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, 1999.
179. G. Marchionini. *Information Seeking in Electronic Environments*. Cambridge University Press, New York, 1997.
180. K. Marriott and P. J. Stuckey. *Programming with Constraints: An Introduction*. MIT Press, Cambridge, 1998.
181. J. Martínez and E. Loisant. Browsing image databases with Galois’ lattices. In *Symp. Applied Computing*. ACM Press, New York, 2002.
182. J. M. Martínez, C. González, O. Fernández, C. García, and J. de Ramón. Towards universal access to content using mpeg-7. In *MULTIMEDIA '02: Proceedings of the Tenth ACM International Conference on Multimedia*. ACM Press, New York, 2002.
183. K. McKeown, R. Barzilay, J. Chen, D. K. Elson, D. K. Evans, J. Klavans, A. Nenkova, B. Schiffman, and S. Sigelman. Columbia’s Newsblaster: New features and future directions. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, 2003.
184. C. Meghini and Y. Tzitzikas. An abduction-based method for index relaxation in taxonomy-based sources. In *Proceedings of MFCS 2003, 28th International Symposium on Mathematical Foundations of Computer Science*, Bratislava, Slovak Republic, 2003. Springer, Berlin, 2003.
185. C. Meghini and Y. Tzitzikas. Querying Articulated Sources. In *Proceedings of the 3rd Intern. Conf. on Ontologies, Databases and Applications of Semantics for Large Scale Information Systems (ODBASE'2004)*, 2004.
186. C. Meghini, Y. Tzitzikas, and A. Analyti. Query evaluation in P2P systems of taxonomy-based sources: algorithms, complexity, and optimizations. CoRR, 0709.3034, 2007.

187. C. Meghini, Y. Tzitzikas, and N. Spyrtos. Abduction for accessing information sources. *Fundamenta Informaticae*, 83(4), 2008.
188. Mercado, <http://www.mercado.com>.
189. D. R. Millen, J. Feinberg, and B. Kerr. Dogear: Social bookmarking in the enterprise. In R. E. Grinter et al., editors, *Conf. Human Factors in Computing Systems (CHI)*. ACM Press, New York, 2006.
190. G. A. Miller. The magic number seven plus or minus two. *Psychological Review*, 63:81–97, 1956.
191. J. Mills and V. Broughton. *Bliss Bibliographic Classification*. Bowker-Saur, London, 1970.
192. D. Mladenčić. Personal web watcher: design and implementation. Technical report, J. Stefan Institute, Department for Intelligent Systems, Ljubljana, Slovenia, 1998.
193. Moodle, <http://www.moodle.org>.
194. B. M. E. Moret. Decision trees and diagrams. *ACM Computing Surveys*, 14(4):593–623, 1982.
195. M. Narayan, C. Williams, S. Perugini, and N. Ramakrishnan. Staging transformations for multimodal web interaction management. In *Proc. of WWW'04*, 2004.
196. L. Nourine and O. Raynaud. A fast incremental algorithm for building lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14:217–227, 2002.
197. M. A. Olson, K. Bostic, and M. Seltzer. Berkeley DB. In *FREENIX Track: USENIX Annual Technical Conf.* USENIX Association, Berkeley, 1999.
198. E. J. O'Neil, P. E. O'Neil, and G. Weikum. The LRU-K page replacement algorithm for database disk buffering. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC, May 26–28, 1993. ACM Press, New York, 1993.
199. P. E. O'Neil. Model 204 architecture and performance. In *Proceedings of the 2nd International Workshop on High Performance Transaction Systems*, London, UK, 1989. Springer, Berlin, 1989.
200. E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for RDF data. In *ISWC*, 2006.
201. Orphanet. An online database on rare diseases and orphan drugs. ©INSERM 1997, <http://www.orpha.net>.
202. Y. Padioleau and O. Ridoux. A logic file system. In *Usenix Annual Technical Conference*. USENIX Association, Berkeley, 2003.
203. Y. Padioleau and O. Ridoux. A parts-of-file file system. In *USENIX Annual Technical Conference, General Track (Short Paper)*, 2005.
204. Y. Padioleau, B. Sigonneau, and O. Ridoux. LISFS: a logical information system as a file system. In L. J. Osterweil, H. D. Rombach and M. L. Soffa, editors, *ICSE*. ACM Press, New York, 2006.
205. C. R. Palmer, J. Pesenti, R. E. Valdes-Perez, M. G. Christel, A. G. Hauptmann, D. Ng, and H. D. Wactlar. Demonstration of hierarchical document clustering of digital library retrieval results. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM Press, New York, 2001.
206. P. Papadakos, Y. Theoharis, Y. Marketakis, N. Armenatzoglou, and Y. Tzitzikas. Mitos: design and evaluation of a DBMS-based web search engine. In *Proc. of 12th Pan-Hellenic Conference of Informatics (PCI'2008)*, Greece, August 2008.
207. P. Papadakos, G. Vasiliadis, Y. Theoharis, N. Armenatzoglou, S. Kopidaki, Y. Marketakis, M. Daskalakis, K. Karamaroudis, G. Linardakis, G. Makrydakos, V. Papatthanasiou, L. Sardis, P. Tsialiamanis, G. Troullinou, K. Vandikas, D. Velegrakos, and Y. Tzitzikas. The anatomy of Mitos web search engine. CoRR, Information Retrieval, <http://arxiv.org/abs/0803.2220>, 2008.
208. P. Papadakos, S. Kopidaki, N. Armenatzoglou, and Y. Tzitzikas. Exploratory web searching with dynamic taxonomies and results clustering. In *Proceedings of the 13th European Conference on Digital Libraries, ECDL'09*, Corfu, Greece, September 2009.

209. A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of the KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
210. J. W. Payne, J. R. Bettman, and E. J. Johnson. *The Adaptive Decision Maker*. Cambridge University Press, New York, 1993.
211. S. Perugini and N. Ramakrishnan. Personalizing web sites with mixed-initiative interaction. *IEEE IT Professional*, 5(2):9–15, 2003.
212. P. Pirolli and S. Card. Information foraging models of browsers for very large document spaces. In *Proceedings of the Advanced Visual Interfaces Workshop, AVI '98*, pp. 83–93, 1998.
213. A. S. Pollitt. The key role of classification and indexing in view-based searching. *International Cataloguing and Bibliographic Control*, 27(2):37–40, 1998.
214. D. Poole. Representing knowledge for logic-based diagnosis. In *Int. Conf. Fifth Generation Computer Systems*. Springer, Berlin, 1988.
215. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
216. R. Prieto-Diaz. Implementing faceted classification for software reuse. *Communications of the ACM*, 34(5):88–97, 1991.
217. U. Priss. Formal concept analysis in information science. *Annual Review of Information Science and Technology*, 40:521–543, 2006.
218. U. Priss. Facet-like structures in computer science. *Axiomathes*, 18:242–255, 2008.
219. Knowledge Processors. Digital Camera demo, <http://www.knowledgeprocessors.com/frame.asp?ID=11>.
220. Knowledge Processors. Universal Knowledge Processor, <http://www.knowledgeprocessors.com>.
221. Knowledge Processors. Wine data mining demo, <http://www.knowledgeprocessors.com/frame.asp?ID=25>.
222. J. O'Regan, R. Rensink and J. Clark. To see or not to see: The need for attention to perceive changes in scenes. *Psychological Science*, 8:368–373, 1997.
223. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema mapping. *VLDB Journal*, 10(4):334–350, 2001.
224. S. R. Ranganathan. *The Colon Classification*. Rutgers University Press, New Brunswick, 1965.
225. S. R. Ranganathan. The colon classification. In S. Artandi, editor, *Rutgers Series on Systems for the Intellectual Organization of Information*, vol. IV. Rutgers University Press, New Brunswick, 1965.
226. S. R. Ranganathan. *Prolegomena to Library Classification*. Asia Publishing House, Bombay, 1967.
227. J. A. Reggia, D. S. Nau, and P. Y. Wang. Diagnostic expert systems based on a set covering model. In M. J. Coombs, editor, *Developments in Expert Systems*, pages 35–58. Academic Press, London, 1984.
228. R. E. Rice, M. Mccreadie, and S.-J. L. Chang. *Accessing and Browsing Information and Communication*. MIT Press, Cambridge, 2001.
229. S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.
230. A. Rockley. *Managing Enterprise Content: A Unified Content Strategy*. New Riders Press, 2002.
231. G. M. Sacco. Dynamic taxonomy process for browsing and retrieving information in large heterogeneous data bases, US Patent 6,763,349.
232. G. M. Sacco. Dynamic taxonomy process for browsing and retrieving information in large heterogeneous data bases, US Patent 7,340,451.
233. G. M. Sacco. Dynamic taxonomy process for browsing and retrieving information in large heterogeneous data bases, US Patent Application 20080133490.
234. G. M. Sacco. Navigating the CD-ROM. In *Proc. Int. Conf. Business of CD-ROM*, 1987.
235. G. M. Sacco. The fact model: a semantic data model for complex databases. *Information Systems*, 13(1):1–11, 1988.

236. G. M. Sacco. Dynamic taxonomies: a model for large information bases. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):468–479, 2000.
237. G. M. Sacco. Process for the assisted comparison of objects described by a set of characteristics, US Patent Application, 2001.
238. G. M. Sacco. Dynamic taxonomies and guided searches. In *ACM CHI 2003—Best Practices and Future Visions for Search UIs: A Workshop*, Miami, USA, 2003.
239. G. M. Sacco. The intelligent e-sales clerk: the basic ideas. In *IFIP INTERACT'03*, Zürich, Switzerland, September 1–5, 2003.
240. G. M. Sacco. Systematic browsing for multimedia infobases. In *Proceedings of the International Conference on Imaging Science, Systems and Technology, CISST '03*, June 23–26, 2003, Las Vegas, Nevada, USA, 2003.
241. G. M. Sacco. Uniform access to multimedia information bases through dynamic taxonomies. In *IEEE Sixth Intern. Symp. on Multimedia Software Engineering (ISMSE'04)*, 2004.
242. G. M. Sacco. DBWorld Xtended: semantic dissemination of information through dynamic taxonomies. In *5th Int. Conf. on Knowledge Management, I-KNOW05*, Graz, Journal of Universal Computer Science, 2005.
243. G. M. Sacco. Guided interactive diagnostic systems. In *18th IEEE Symposium on Computer-Based Medical Systems (CBMS 2005)*, 23–24 June 2005, Dublin, Ireland, 2005.
244. G. M. Sacco. Guided interactive information access for e-citizens. In *Proc. of the 4th Intern. Conf. on Electronic Government (EGOV-2005)*, 2005.
245. G. M. Sacco. The intelligent e-store: Easy interactive product selection and comparison. In *Proceedings of the 7th IEEE Intern. Conf. on E-Commerce Technology (CEC-2005)*, 2005.
246. G. M. Sacco. No (e-)democracy without (e-)knowledge. In *Proceedings of International Conference on E-Government: Towards Electronic Democracy, TCGOV 2005*, Bolzano, Italy, March 2–4, 2005.
247. G. M. Sacco. Analysis and validation of information access through mono, multidimensional and dynamic taxonomies. In *Proceedings of 7th International Conference on Flexible Query Answering Systems, FQAS 2006*, Milan, Italy, June 7–10, 2006. LNCS, vol. 4027. Springer, Berlin, 2006.
248. G. M. Sacco. Some research results in dynamic taxonomy and faceted search systems. In *SIGIR'2006 Workshop on Faceted Search*, Seattle, WA, USA, August 2006.
249. G. M. Sacco. User-centric access to e-government information: E-citizen discovery of E-services. In *2006 AAAI Spring Symposium Series*, Stanford University, 2006.
250. G. M. Sacco. E-commerce portals: guided product selection and comparison. In A. Tatnall, editor, *Encyclopedia of Portal Technologies and Applications*. IGI Global, Hershey, 2007.
251. G. M. Sacco. Interactive exploration and discovery of e-government services. In *dg.o 2007: Proceedings of the 8th Annual International Conference on Digital Government Research*. Digital Government Society of North America, Philadelphia, 2007.
252. G. M. Sacco. DT-miner: Data mining for the people. In *DEXA FIND'08, 2nd Int. Workshop on Dynamic Taxonomies and Faceted Search*. IEEE Computer Society, Los Alamitos, 2008.
253. G. M. Sacco. Dynamic taxonomies: Guided interactive diagnostic assistance. In E. Geisler and N. Wickramasinghe, editors, *Encyclopedia of Healthcare Information Systems*. IGI Global, Hershey, 2008.
254. G. M. Sacco. e-RARE: Interactive diagnostic assistance for rare diseases through dynamic taxonomies. In *DEXA FIND'08, 2nd Int. Workshop on Dynamic Taxonomies and Faceted Search*. IEEE Computer Society, Los Alamitos, 2008.
255. G. M. Sacco. Intelligent user-centric access to public information. In M. Khosrow-Pour and G. D. Garson, editors, *Handbook of Research on Public Information Technology*. IGI Global, Hershey, 2008.
256. G. M. Sacco. Rosso Tiziano: A system for user-centered exploration and discovery in large image information bases. In *DEXA 2008, 19th International Conference on Database and Expert Systems Applications*. LNCS. Springer, Berlin, 2008.
257. G. Sacco. Correspondence with the ACM Digital Library group, 2000.
258. G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, 1971.

259. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
260. M. Sanderson and W. B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*, 1999.
261. S. Santini and R. Jain. Integrated browsing and querying for image databases. *IEEE Multimedia*, 7:26–39, 2000.
262. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system—a case study. In *Proceedings of the Workshop on Web Mining and E-Commerce Systems at KDD 2000 (WEBKDD 2000)*, August 2000.
263. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on the World Wide Web (WWW '01)*, 2001.
264. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT 2002)*, 2002.
265. J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1–2):115–153, 2001.
266. A. Th. (Guus) Schreiber, B. Dubbeldam, J. Wielemaker, and B. Wielinga. Ontology-based photo annotation. *IEEE Intelligent Systems*, 16(3):66–74, 2001.
267. B. Séroussi, J. Bouaud, and É.-C. Antoine. Oncodoc: a successful experiment of computer-supported guideline development and implementation in the treatment of breast cancer. *Artificial Intelligence in Medicine*, 22(1):43–64, 2001.
268. X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*. ACM Press, New York, 2005.
269. H. Shimazu. Expertclerk: A conversational case-based reasoning tool for developing sales-clerk agents in e-commerce webshops. *Artificial Intelligence Review*, 18(3–4):223–244, 2002.
270. B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.
271. V. Sinha and D. R. Karger. Magnet: Supporting navigation in semistructured data environments. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*, 2005.
272. A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.
273. M. B. Smyth. Power domains. *Journal of Computer and System Sciences*, 16:23–36, 1978.
274. R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006)*, 2006.
275. E. Sperner. Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, 27:544–548, 1928.
276. L. Spiteri. A simplified model for facet analysis. *Canadian Journal of Information and Library Science*, 23:1–30, 1998.
277. R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB '95: Proceedings of the 21st International Conference on Very Large Data Bases*. Morgan Kaufmann, San Francisco, 1995.
278. M. Stefaner and B. Müller. Elastic lists for facet browsers. In *DEXA '07: Proceedings of the 18th International Conference on Database and Expert Systems Applications*. IEEE Computer Society, Washington, 2007.
279. M. Stefaner, T. Urban, and M. Seefelder. Elastic lists for facet browsing and resource analysis in the enterprise. In *DEXA Workshops*. IEEE Computer Society, Los Alamitos, 2008.

280. M. Stefaner, E. D. Vecchia, M. Condotta, M. Wolpers, M. Specht, S. Apelt, and E. Duval. Mace—enriching architectural learning objects for experience multiplication. In *EC-TEL*, 2007.
281. E. Stoica and M. A. Hearst. Nearly-automated metadata hierarchy creation. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*, 2004.
282. E. Stoica, M. A. Hearst, and M. Richardson. Automating creation of hierarchical faceted metadata structures. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007)*, 2007.
283. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International World Wide Web Conference (WWW 2007)*, 2007.
284. J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, New York, 2004.
285. J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *Proceedings of the 31st Annual International Conference on Research and Development in Information Retrieval*, 2008.
286. J. Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, Sebastopol, 2005.
287. R. Tiziano. <http://tiziano.di.unito.it>.
288. D. Tunkelang. Dynamic category sets: An approach for faceted search. In *ACM SIGIR '06 Workshop on Faceted Search*, August 2006.
289. A. Turpin and F. Scholer. User performance versus precision measures for simple search tasks. In *SIGIR '06: Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.
290. M. Tvarožek and M. Bieliková. Personalized faceted navigation in the semantic web. In *Proceedings of the International Conference on Web Engineering (ICWE 2007)*, 2007.
291. B. Tversky and J. B. Morrison. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57:247–262, 2002.
292. Y. T. Tzitzikas. Collaborative ontology-based information indexing and retrieval. PhD thesis, Department of Computer Science, University of Crete, September 2002.
293. Y. Tzitzikas. An algebraic method for compressing symbolic data tables. *Journal of Intelligent Data Analysis (IDA)*, 10(4):343–359, 2006.
294. Y. Tzitzikas. Evolution of faceted taxonomies and CTCA expressions. *Knowledge and Information Systems (KAIS)*, 13(3):337–365, 2007.
295. Y. Tzitzikas and A. Analyti. Mining the meaningful term conjunctions from materialised faceted taxonomies: algorithms and complexity. *Knowledge and Information Systems Journal*, 9(4):430–467, 2006.
296. Y. Tzitzikas and C. Meghini. Ostensive automatic schema mapping for taxonomy-based peer-to-peer systems. In *Proceedings of the 7th Intern. Workshop on Cooperative Information Agents (CIA-2003)*, 2003. (Best Paper Award.)
297. Y. Tzitzikas and C. Meghini. Query evaluation in peer-to-peer networks of taxonomy-based sources. In *Proceedings of 19th Int. Conf. on Cooperative Information Systems, CoopIS'2003*, Catania, Sicily, Italy, November 2003.
298. Y. Tzitzikas and Y. Theoharis. Naming functions for the vector space model. In *Proceedings of the 29th European Conference on Information Retrieval, ECIR'2007*, Rome, Italy, April 2007.
299. Y. Tzitzikas, N. Spyrtos, P. Constantopoulos, and A. Analyti. Extended faceted ontologies. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering, CAiSE-02*, Toronto, Canada, May 2002 (short paper).
300. Y. Tzitzikas, N. Spyrtos, P. Constantopoulos, and A. Analyti. Extended faceted taxonomies for web catalogs. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering, WISE 2002*, Singapore, 2002.

301. Y. Tzitzikas, A. Analyti, N. Spyratos, and P. Constantopoulos. An algebraic approach for specifying compound terms in faceted taxonomies. In *Information Modelling and Knowledge Bases XV, 13th European-Japanese Conference on Information Modelling and Knowledge Bases, EJC'03*. IOS Press, Amsterdam, 2004.
302. Y. Tzitzikas, R. Launonen, M. Hakkarainen, P. Kohonen, T. Leppanen, E. Simpanen, H. Tornroos, P. Uusitalo, and P. Vanska. FASTAXON: a system for FAST (and faceted) TAXONOMY design. In *Proceedings of 23rd Int. Conf. on Conceptual Modeling, ER'2004*, Shanghai, China, November 2004.
303. Y. Tzitzikas, C. Meghini, and N. Spyratos. A unifying framework for flexible information access in taxonomy-based sources. In *Proceedings of the 6th Intern. Conference on Flexible Query Answering Systems, FQAS'2004*, Lyon, France, July 2004.
304. Y. Tzitzikas, C. Meghini, and N. Spyratos. Towards a generalized interaction scheme for information access. In *Proceedings of the 3rd Intern. Symposium on Foundations of Information and Knowledge Systems, FoKS'2004*, Vienna Austria, February 2004.
305. Y. Tzitzikas, A. Analyti, and N. Spyratos. Compound term composition algebra: the semantics. *LNCIS Journal on Data Semantics*, 2:58–84, 2005.
306. Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. Mediators over taxonomy-based information sources. *VLDB Journal*, 14(1):112–136, 2005.
307. Y. Tzitzikas, C. Meghini, and N. Spyratos. A unified interaction scheme for information sources. *Journal of Intelligent Information Systems*, 26(1):75–93, 2006.
308. Y. Tzitzikas, A. Analyti, N. Spyratos, and P. Constantopoulos. An algebra for specifying valid compound terms in faceted taxonomies. *Data & Knowledge Engineering*, 62(1):1–40, 2007.
309. Y. Tzitzikas, N. Armenatzoglou, and P. Papadakos. FleXplorer: A framework for providing faceted and dynamic taxonomy-based information exploration. In *Proc. of FIND'2008 (at DEXA '08)*, Turin, Italy, September 2008.
310. Y. Tzitzikas, N. Armenatzoglou, and P. Papadakos. On providing efficient faceted and dynamic taxonomy-based information exploration services, submitted.
311. C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
312. C. J. van Rijsbergen, F. Crestani and M. Lalmas, editors. *Information Retrieval: Uncertainty and Logics. Advanced Models for the Representation and Retrieval of Information*. Kluwer Academic, Dordrecht, 1998.
313. B. C. Vickery. Knowledge representation: a brief review. *Journal of Documentation*, 42(3):145–159, 1986.
314. Vivisimo. <http://www.vivisimo.com/>.
315. J. R. White. On the 10th anniversary of ACM's Digital Library. *Communications of the ACM*, 51(11):5–5, 2008.
316. G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25:38–49, 1992.
317. W. Willett, J. Heer, and M. Agrawala. Scented widgets: improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007.
318. D. Wollersheim and J. W. Rahayu. Methodology for creating a sample subset of dynamic taxonomy to use in navigating medical text databases. In *International Database Engineering & Applications Symposium, IDEAS'02*, July 17–19, 2002, Edmonton, Canada, 2002.
319. K. Wu, E. Otoo, and A. Shoshani. On the performance of bitmap indices for high cardinality attributes. In *VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases*. Morgan Kaufmann, San Francisco, 2004.
320. S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In *Proceedings of the 31st Annual International Conference on Research and Development in Information Retrieval*, 2008.
321. Yahoo!, <http://www.yahoo.com/>.
322. Yahoo! Shopping, <http://shopping.yahoo.com>.

323. K. Yang, E. K. Jacob, A. Loehrlein, S. Lee, and N. Yu. The best of both worlds: A hybrid approach to the construction of faceted vocabularies. In *Proceedings of the Asian Information Retrieval Symposium*, 2004.
324. Y. Yang, S. Yoo, J. Zhang, and B. Kisiel. Robustness of adaptive filtering methods in a cross-benchmark evaluation. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.
325. S. B. Yao. Approximating block accesses in database organizations. *Communications of the ACM*, 20(4):260–261, 1977.
326. F. Yates. Contingency table involving small numbers and the χ^2 test. *Journal of the Royal Statistical Society (Supplement)*, 1(2):217–235, 1934.
327. K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the Conference on Human Factors in Computing Systems*. ACM Press, New York, 2003.
328. K. Yu, V. Tresp, and S. Yi. A nonparametric hierarchical bayesian framework for information filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*. ACM Press, New York, 2004.
329. L. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28, 1978.
330. O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, August 1998. ACM Press, New York 1998.
331. C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of Tenth International Conference on Information and Knowledge Management*, 2001.
332. J. Zhang and G. Marchionini. Coupling browse and search in highly interactive user interfaces: a study of the relation browser++. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM Press, New York, 2004.
333. L. Zhang, F. Lin, and B. Zhang. A CBIR method based on color-spatial feature. In *TENCON'99*, 1999.
334. Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation systems. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. ACM Press, New York, 2007.
335. D. X. Zhou, N. Oostendorp, M. Hess, and P. Resnick. Conversation pivots and double pivots. In *Computer–Human Interaction (CHI)*. ACM Press, New York, 2008.
336. P. Zigoris and Y. Zhang. Bayesian adaptive user profiling with explicit and implicit feedback. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM 2006)*. ACM Press, New York, 2006.
337. G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Reading, 1949.
338. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23:337–343, 1977.

Index

A

A-Box, 67, 134
Abduction, 174
Absolute path, 294
Abstract data type, 254
Abstraction, 131, 255
Access control, 139
Adaptation, 161
Advanced search, 76
Algebra, 147
Art collections, 264
Aspect, 23
Assertion, 68
Association rules, 114
Attribute, 59
Attribute–value pairs, 25
Axiom, 68

B

Backwards inheritance, 7, 219
Balanced dynamic taxonomies, 178
Bibliography, 296
Bit vectors, 222
Boolean information retrieval, 41
Boolean query, 78
Boolean querying, 294

C

Caching of related sets, 234
Camelis, 79
CBIR, 272
Centroid, 43
Change blindness, 78
Chi square test, 116
Classification, 69
Classification statement, 133
Classifier, 193
 evaluation, 203
 training, 194
Click logs, 103
Cluster
 centroid, 43, 278
Cluster hypothesis, 44
Clustering, 43, 60, 141, 274
 monodimensional vs.
 multidimensional, 275

 results, 246
 Suffix Tree Clustering, 246
Collaborative filtering, 104
Color space
 CIE L·a·b, 277
 HSV, 277
 RGB, 277
Command-line interface, 296
Complete description, 25
Completeness, 255
Completeness of iteration, *see* self-adapting
 exploration structures, completeness
 of iteration
Completeness of reach, *see* self-adapting
 exploration structures, completeness
 of reach
Component, 254
Composition, 256
Compound ordering, 148
Compound taxonomy, 247
Compound term, 25, 146
Compound terminology, 147
Concept, 6, 7, 60, 66
 extension, 69
 extent, 59
 identifier, 217
 intent, 59
 label, 179
 lattice, 60
 ordering, 69
 time-varying, 236
Concepts, 65
Concrete domain, 256
Confidence, 114, 116
Consistency, 255
Constructor, 66
Content-based image retrieval, 272
Cosine similarity, 42
Count strategy, 220
Cross aspect, 176
CTCA, 145, 247
 application, 247
 expression mining, 159
 materialized faceted taxonomy, 159
 revision, 155
 semantics, 155

Cultural heritage, 264
 Customization, 132, 253

D

Data mining, 114
 Database
 ORDBMS, 240
 relational, 237, 248
 SQL, 237
 view, 237
 Datalog, 174
 Decision procedure, 255
 Decision tree, 57
 Deep extension, 6, 24, 219
 Derived virtual concepts, 235
 Description
 complete, 25
 Description function, 130
 Description Logics, 66, 133, 258
 semantics, 67
 SHOIQ, 67
 signature, 66
 Dewey classification, 12
 Diagnostic systems, 282
 Digital libraries, 288
 Disjunction, 64, 84
 Drill-down, 40
 DT, 5
 Dynamic taxonomy, 5, 8

E

E-auctions, 265
 E-commerce, 265
 end game, 266, 269
 enhanced feature display, 270
 product selection, 265
 thinning game, 266
 weighted additive strategy, 268
 E-government, 290
 laws and regulations, 290
 local promotion, 291
 services, 291
 E-health, 264
 E-hrm, 264
 E-learning, 264
 E-mail, 296
 E-matchmaking, 264
 E-recruitment, 264
 Embeddability, 253
 Endgame, 75
 Entity–Relationship model, 180
 Evolution, 155
 Expected confidence, 116
 Exploration model, 37

Exploratory patterns
 knowledge-seeking tasks, 4
 object-selection tasks, 3, 263
 wisdom-seeking, 4, 114, 263, 289
 Exploratory search, 1, 62
 Expressivity, *see* self-adapting exploration
 structures, expressivity, 66
 Expressivity/efficiency trade-off, 253
 Extension, 70, 131, 133
 deep, 6, 24, 28, 219
 shallow, 6, 24, 28, 219
 Extensional, 145, 155
 Extensional inference rule, 7, 8
 Extensionally related terms, 31
 Extent, 60
 Extrinsic property, 295

F

Facet, 12, 21
 analysis, 23
 discovery, 196
 extraction, 195
 free-text, 108
 interval, 108
 nominal, 108
 ordinal, 108
 orthogonality, 19, 24
 ratio-typed, 108
 types, 25, 108
 Facet Analysis
 Canon of Concomitance, 176
 fundamental categories, 186
 Principle of Ascertainability, 177
 Principle of Division, 176
 Principle of Mutual Exclusion, 176
 Principle of Permanence, 177
 Principle of Relevant Succession, 179
 Facet classification, 194
 Facet orthogonality, 176
 Facet–value pairs ranking, 104
 Faceted search, 5, 15
 personalized, 103
 Faceted taxonomy, 19
 Fact table, 39
 FASTAXON, 247
 FCA, 59
 File, 294
 File system, 294
 Filtering, 40
 FleXplorer, 245
 Focalized search, 1
 Focus, 8, 28, 62, 245
 extensional, 28

- intensional, 28
- redundancy free, 28
- Formal concept, 60
- Formal concept analysis, 59
- Formal context, 59
- Formats, 250
- Full-text search, 296
- Fundamental facets, 175
- Fuzzy dynamic taxonomies, 138, 284
- Fuzzy object descriptions, 251
- Fuzzy sets, 138

G

- Galois connection, 60
- Geographic map, 140, 141
- Geographical information system, 298
- GEOLIS, 301
- Global-as-view, 165
- Graphical interface, 296, 300
- Guided navigation, 9
- Guided thinning, 9

H

- Heterogeneous, 297
- Home directory, 297
- Hypermedia, 4
- Hypernym, 193

I

- Index expansion, 143
- Index improvement, 174
- Indexing, 295
- Individual, 66
- Infimum, 60
- Infobase, 2
- Information base, 2
- Information filtering, 103
- Information retrieval, 1, 62, 75, 103, 245
- Information scent, 77
- Initial constraint definition, 75
- Instantiation, 69
- Instantiation statement, 133
- Integration
 - global-as-view, 165
 - local-as-view, 165
 - mappings, 163
 - materialized approach, 172
 - mediator, 165
 - P2P, 169
 - virtual approach, 172
- Intensional, 145, 155
- Intent, 60
- Interaction
 - modeling, 28

- Interest focus, 8
- Interpretation, 24, 67
 - model, 24, 238
 - ordering, 24
- Interval, 26, 129
 - existential meaning, 26
 - query answering, 27
 - universal meaning, 26
- Intervals, 238, 254
- Intrinsic property, 294
- Invalid compound terms, 147
- Inverted lists, 222

J

- Jaccard coefficient, 42
- JSR-170, 252

K

- Knowledge base, 67, 133
 - assertional part, 67
 - terminological part, 67
- Knowledge-seeking tasks, 4

L

- Labor-intensive strategy, 243
- Latent semantic indexing (LSI), 106
- Layer, 298
- Lift, 114
- LISFS, 294, 301
- Literary warrant, 179
- Local view, 133
- Local-as-view, 165
- Logic, 127, 253, 254, 294, 298
 - Description Logics, 66
 - entailment, 127
- Logic functor, 256
- Logic programming, 174
- Logical inference, 69
- Logical information systems, 253, 294, 298
- Logics composition, 253
- Logics engineering, 260
- Low-level multimedia feature, 272

M

- Machine learning, 192
- Map, 300
- Mapping
 - construction, 174
 - ostensive, 163
 - protocol, 165
 - query-to-query, 174
 - term-to-query, 168, 174
 - term-to-term, 168, 174

- Mappings, 169, 174
 - inter-taxonomy, 170
- Market basket model, 114
- Materialized faceted taxonomy, 19, 28, 31, 159
 - restriction, 33
- Materialized integration approach, 172
- Maximum resolution of the taxonomy, 47
- Mediator, 165, 169, 172
- Medical diagnosis, 283
- Meta-data, 2, 295
- Minimal model, 24
- Mining, 159
- Mitos, 245
- Model, 69
 - minimal, 24
- Model interpretation, 24
- Multidimensional taxonomy, 5
- Multimedia infobases, 272
- Multiple inheritance, 177
- Music, 295

- N**
- Named-entity
 - tagger, 197
- Naming function, 163
- Natural language, 84
- Navigation, 62
- Navigation link, 79
- Navigation mode, 78, 136
 - pivot, 83
 - range selection, 85
 - shift, 82
 - slice and dice, 84
 - zoom-in, 81
 - zoom-out, 82
- Navigation space, 63
- Navigation tree, 154, 247
- Negation, 64
- News systems, 288
- No-count strategy, 220
- No-zero-result, *see* self-adapting exploration structures, no-zero-result

- O**
- Object, 5, 59, 66
- Object cluster, 140
- Object identifier, 217
- Object selection, 137
- Object-selection tasks, 3, 263
- OLAP, 39–41, 205, 238
- Ontological similarity, 110
- Ontologies, 65
- Ontology, 66, 109, 133, 198
- ORBMS, 240

- Order
 - partial, 24
 - Smyth, 148
- Orienteering, 75, 77
- Orthogonal subtaxonomies, 12
- OWL, 64, 65, 133, 250
- OWL Lite, DL, Full, 66

- P**
- P2P, 169, 172
- Parallel architectures, 235
- Part-of-file, 294
- Partial on-demand loading, 246
- Partial order, 24
- Partitioning aspect, 176
- Path, 294
- Peer-to-peer, 169, 172
- Personal information management, 298
- Personalization, 139
- Personalized faceted search, 103
- Pivot, 83, 137, 294
- Pivoting, 40
- Poka-yoke, 91
- Poka-yoke principle, 76
- Popularity, 142
- Precision, 43
- Predefined focus, 139
- Principle of Conservation, 156
- Principle of Persistence of Prior Knowledge, 156
- Prolog, 71
- Protocols, 252

- Q**
- Query, 69, 78, 133
- Query evaluation
 - distributed, 172
- Query expansion, 46, 143
- Query reversal, 137
- Querying, 62
- Querying-by-example, 85

- R**
- Range selection, 85
- RDBMS, 237
- RDF, 65, 250
- RDF Schema, 65, 250
- RDF/S, 64
- Recall, 43
- Reduced taxonomy, 8, 33
 - computation, 220
 - count strategy, 225
 - focus-driven, 220
 - full-loading strategy, 221

- labor-intensive strategy, 221
- lazy strategy, 222
- no-count strategy, 225
- on-demand partial loading strategy, 222
- taxonomy-driven, 221
- Reference view, 122
- Related count, 8
- Related queries, 45
- Related set of concepts, 8
- Relation, 66
- Relational database, 297
- Relational view, 181
- Relative path, 294
- Relevance feedback, 103
- Repository, 217
- Restriction of a materialized faceted taxonomy, 33
- Results clustering, 246
- Retrieval model, 35
- Reward function, 111
- Role, 66
- Role traversal, 138
- Roll-up, 40
- RQL, 70
- S**
- SAES, *see* self-adapting exploration structures
- Scatter–Gather, 44, 276
- Search engine, 242
 - Endeca, 243
 - faceted exploration, 245
 - Google Base, 243
 - i411, 243
 - Knowledge Processors, 243
 - Mercado, 243
 - metadata, 244
 - Mitos, 245
 - Siderean Seamark, 243
 - Solr, 243
- Secondary focus, 121
- Security and privacy, 139
- Selection, 79
- Selector, 36, 37
- Self-adapting exploration, 13
- Self-adapting exploration structures, 37
 - completeness of iteration, 37
 - completeness of reach, 37
 - expressivity, 37
 - no-zero-result, 37
 - self-sufficiency, 37
- Self-sufficiency, *see* self-adapting exploration structures, self-sufficiency
- Semantic web, 64, 133
 - exploratory search, 70
 - ontologies, 65
 - OWL, 64
 - RDF/S, 64
 - RQL, 70
 - SPARQL, 70
- Semantics, 24, 27, 67, 255
- Sense disambiguation, 193
- Shallow extension, 6, 24, 219
- Shift, 82
- Signature, 66, 255
- Similarity function, 42
- Singular value decomposition, 105, 106
- Skyline, 271
- Slice-and-dice, 40, 84, 205
- Smyth order, 148
- Snippet, 45, 246
- SPARQL, 70
- Sperner system, 154
- SQL, 237
- String pattern, 129
- Structured objects, 123
- Subsumption, 5, 24, 69, 70, 128, 255
- Suffix Tree Clustering, 246
- Support, 114
- Support Vector Machines (SVM), 203
- Supremum, 60
- Symbolic data tables, 160
- Syntax, 66, 255
- Systems
 - architecture, 241
 - Semantic Web, 71
- T**
- T-Box, 67, 136
- Tag cloud, 140
 - extended, 38
- Taxonomy, 19, 20, 127, 134
 - analysis of pruning, 52
 - as a logic, 128
 - automatic construction, 202, 238, 246
 - change, 156
 - composition, 146, 253
 - compound, 149, 247
 - convergence, 46
 - depth, 238
 - derived, 132
 - dynamic, 8, 33
 - evolution, 155
 - faceted, 19
 - implementation, 248
 - implicitly-defined, 128
 - infinite, 128
 - mappings, 163
 - materialized, 19

- materialized faceted, 28
- maximum resolution, 47
- monodimensional, 47
- multidimensional, 5, 47
- reduced, 8, 33, 34
- retrofit of monodimensional, 189
- Taxonomy-based source, 24
- Term, 19
 - compound, 25, 146
 - invalid, 147
 - valid, 147
 - concept, 19
 - discovery, 196
 - extensionally related, 31
 - extraction, 197
- Terminology, 20, 24
 - compound, 147
- Test view, 122
- Text classification, 194
- Text-annotated collections, 192
- Thesaurus, 143
- Transducer, 294

- U**
- UML, 241
 - component diagram, 241
 - sequence diagram, 243
- User feedback, 103, 161
 - implicit, 103
- User preferences, 104

- User ratings, 104
- User relevance model, 108

- V**
- Valid compound terms, 147
- Vector-space information retrieval, 41
- View, 71
- Virtual concept, 126, 178
 - derived, 126
 - implementation, 235
 - simple, 126, 235
- Virtual integration approach, 172

- W**
- Wikipedia, 198
- Wisdom-seeking, 4, 114, 263, 289
- WordNet, 193

- X**
- XFML, 250
- XML, 65, 250
- XML Schema, 65, 250

- Z**
- Zoom, 9, 137, 218, 301
 - in, 30, 81, 246, 294
 - out, 32, 82, 243
 - ranking, 30
 - side, 31, 171
- Zoom point, 29

Appendix A

Color Images

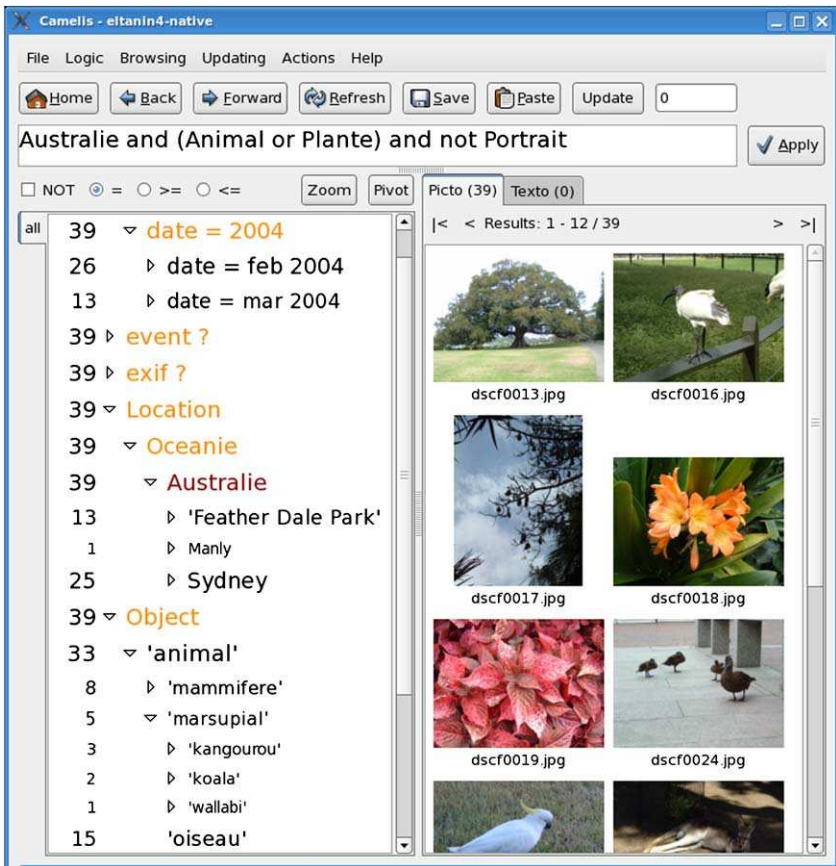


Fig. 4.2C The graphical interface of CAMELIS



Fig. 4.5C The ContentLandscape application (see Sect. 4.5.3) combines bar chart representations with slider controls for range selection

Fig. 4.8C The ContentLandscape application applies the collapsible panel pattern for zooming into concepts within a hierarchy

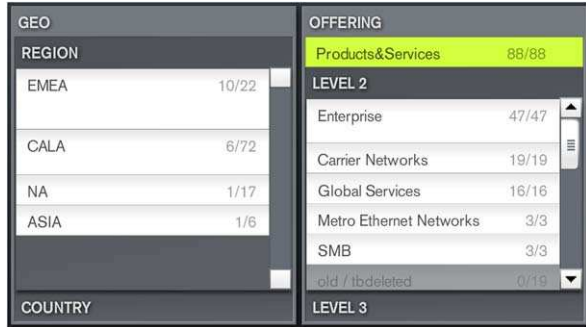


Fig. 4.13C Quick access to concepts with a combo box in the ContentLandscape application





Fig. 4.15C The RAVE system visualizes metadata value proportions in horizontal bar charts

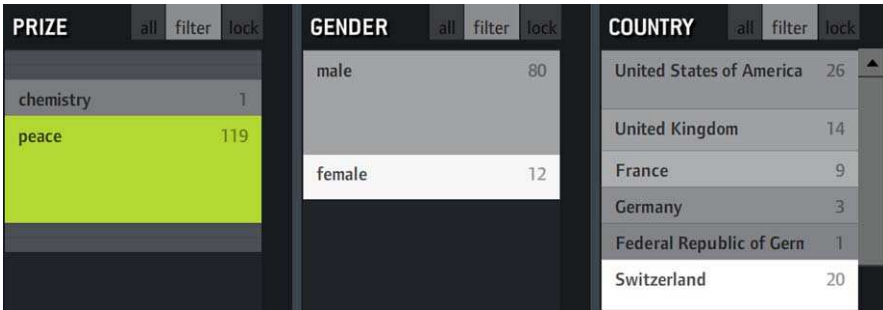
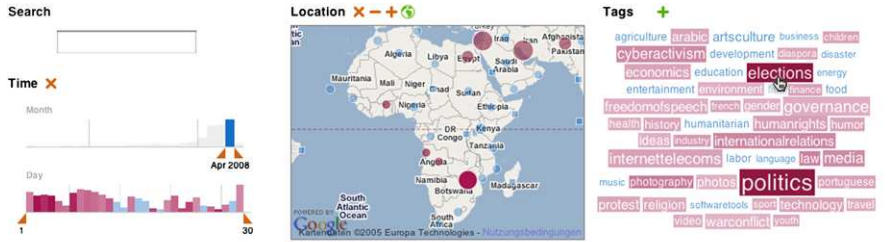


Fig. 4.16C Elastic lists indicate the number of matched resources in scaling list entry height. Additionally, unusually high proportions (compared to the global distribution) are indicated by brightness of the list entries

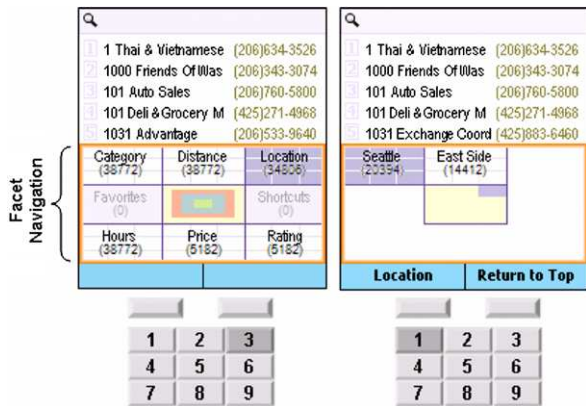


Results 56 of 442



Fig. 4.17C Weighted, coordinated brushing in the visgets system

Fig. 4.18C Faceted search for small screens in the FaThumb prototype



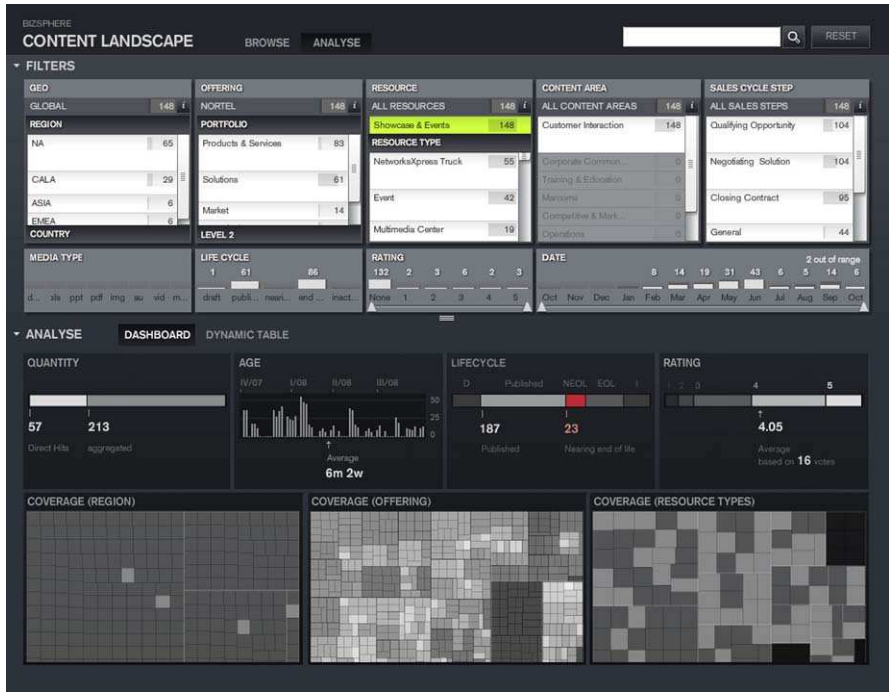


Fig. 4.20C The dashboard view of the ContentLandscape application

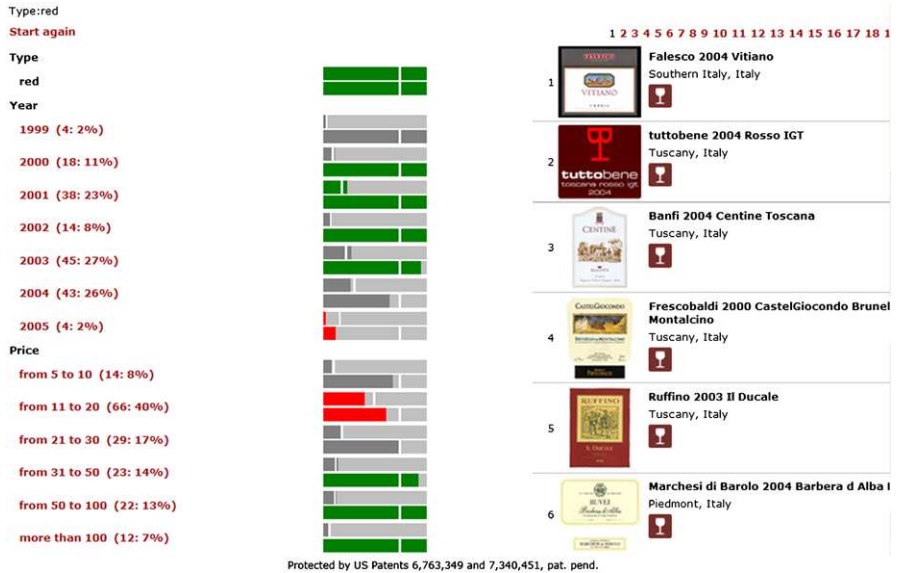


Fig. 5.3C The Italian wines infobase, after a zoom on Red wines

Fig. 5.4C The Italian wines infobase, after a zoom on Red wines and on wines costing more than \$100

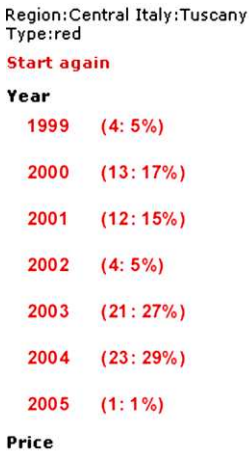
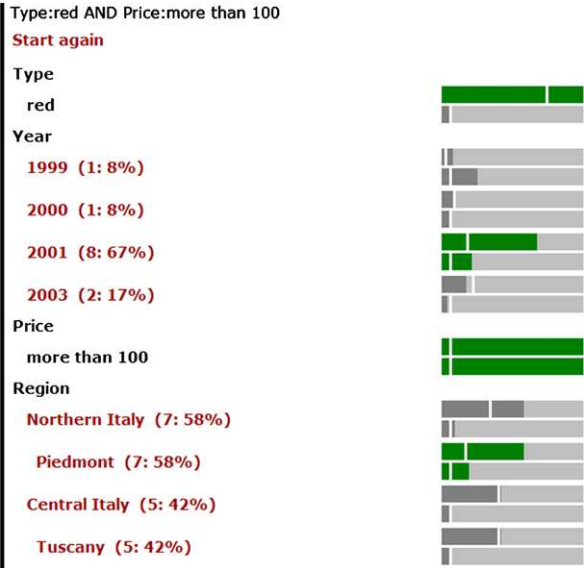


Fig. 5.5C Side-by-side comparison on the Italian wines infobase



Fig. 9.3C Enhanced feature display for Nikon digicams

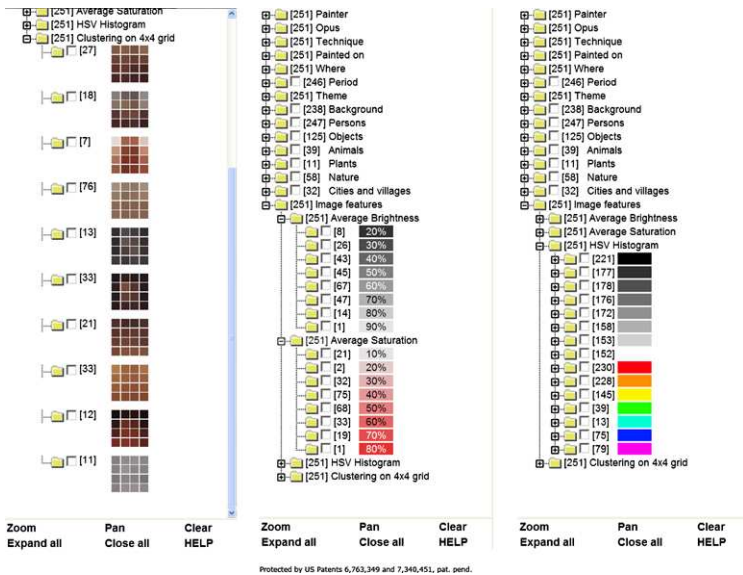


Fig. 9.4C From left: **a** Multidimensional primitive features: clustering of average color on a 4×4 grid. Clusters are labeled by their barycenter; **b** Monodimensional primitive features: average brightness and average saturation; **c** Bidimensional primitive features: reduced HSV histogram

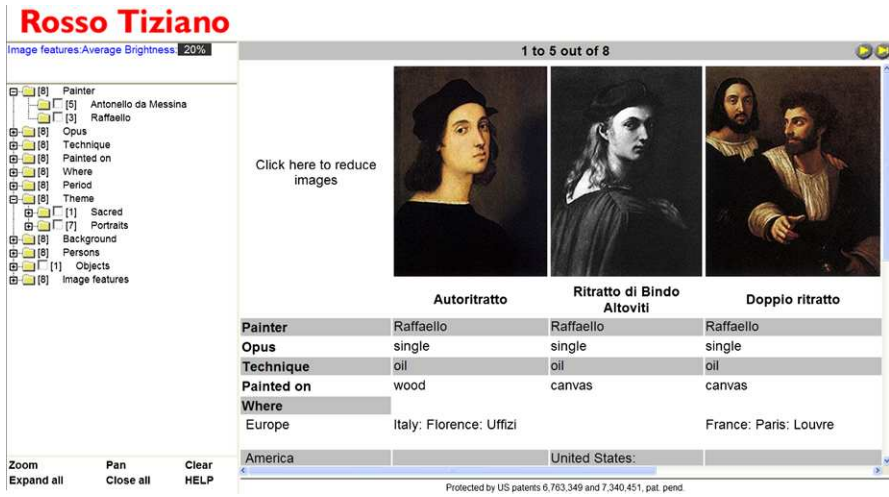


Fig. 9.5C Exploring dark paintings: only Raphael and Antonello have dark items, and almost all are portraits. Dark portraits are expanded

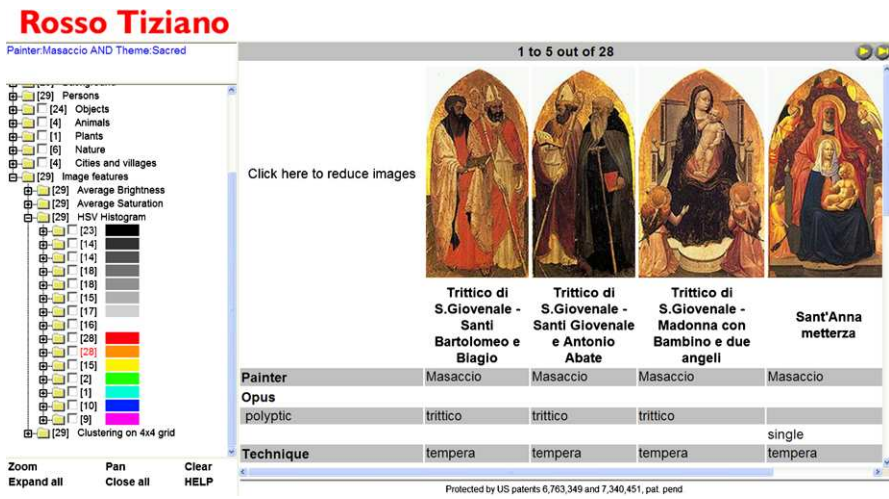







Fig. 9.6C Histogram summary of Masaccio's sacred paintings: paintings with orange-ish colors are displayed

Rosso Tiziano

Painter: Antonello da Messina AND
Theme: Portraits 6 to 10 out of

- [16] Painter
- [16] Opus
- [16] Technique
- [16] Painted on
- [16] Where
- [16] Period
- [16] Theme
- [16] Background
- [16] Persons
- [2] Objects
- [2] Plants
- [16] Image features
 - [16] Average Brightness
 - [16] Average Saturation
 - [16] HSV Histogram
 - [16] Clustering on 4x4 grid
 - [2] 
- [12] 
- [2] 

Click here to reduce images



Ritratto di giovane uomo

Ritratto d'uomo

Painter	Antonello da Messina	Antonello da Messina
Opus	single	single
Technique	oil	oil
Painted on	wood	wood
Where	Europe	Italy: Rome: Galleria Borghese

Zoom
Expand all

Pan
Close all

Clear
HELP

Protected by US patents 6,763,349 and

Fig. 9.7C Cluster summary of Antonello's portraits: displaying the selected cluster

Università di Torino
Dipartimento di Informatica

e-RARE

interactive assisted diagnosis for rare diseases

Database and Information Systems Research Group

Total: 137

QuickSymptom Clear Italiano HELP

Systems:Neurological functional anomalies:Ataxia/incoordination

- 124 Head
- 31 Trunk
- 55 Limbs
- 137 Systems
 - 15 Skeletal anomalies
 - 37 Skin abnormal
 - 24 Hair anomalies
 - 5 Nails anomalies
 - 13 Abnormal alimentary tract
 - 18 Abnormal respiratory system
 - 17 Abnormal urinary tract/renal fur
 - 25 Abnormal genitalia
 - 14 Cardiac anomalies
 - 7 Vascular anomalies
 - 14 Liver/biliary tract anomalies
 - 9 Spleen/pancreas anomalies
 - 5 Immune system anomalies
 - 7 Haematological system
 - 12 Endocrine glands anomalies
 - 32 Metabolism anomalies
 - 1 Phosph./calcium metabolism
 - 1 Hyperammonaemia
 - 3 Organic acids disorders
 - 7 Hyperglycemia/diabetes mel
 - 3 Hyperlipidaemia/hyperchole
 - 3 Lipidosis/sulfatidosis
 - 2 Hypericemia
 - 8 Metabolism of aminoacids a
 - 1 Oligoelements metabolism i
 - 45 Neurological structural anomali
 - 137 Neurological functional anomali
 - 32 Muscle anomalies
 - 4 Neoplasia/cancer
 - 56 Build/stature/survival anomalies
 - 2 Antenatal period anomalies
 - 96 Chromosomal or genetic anomaly

1 to 10 out of 137

- 2906 Poikilo-dermatomyositis
- 2891 Pili torti - developmental delay - neurological abnormalities
- 2954 Mental retardation short broad thumbs Meinecke syndrome
- 3041 Mental retardation balding patella luxation acromicria Scholte beeger van essen syndrome
- 2822 Spastic paraplegia mental retardation corpus callosum thin Nakamura osame syndrome
- 2826 Spastic paraplegia - precocious puberty
- 2802 Anaemia, sideroblastic, X-linked - ataxia
- 3178 Spinocerebellar degeneration book type
- 3177 Corneal-cerebellar syndrome Der Kaloustian-Jarudi-Khoury syndrome Spinocerebellar degeneration - corneal dystrophy
- 3048 Mental retardation hypocupraemia hypobetalipoproteinaemia

Research project by *Dynamic Taxonomies Research Group* Rare disease database by *Orphanet* UKP software donated by Knowledge Processors US patents 6,763,349 and 7,340,451, pat. pend.

Fig. 9.9C Reduced taxonomy and diseases after a zoom on ataxia

Università di Torino
Dipartimento di Informatica

e-RARE

interactive assisted diagnosis for rare diseases

Database and Information Systems Research Group

Total: 7

QuickSymptom Clear Italiano HELP

- 7 Head
- 1 Trunk
- 2 Limbs
- 7 Systems
- 1 Neoplasia/cancer
- 3 Build/stature/survival anomalies
- 6 Chromosomal or genetic anomaly
- 7 Age of onset

Systems:Neurological functional anomalies:Ataxia/incoordination
Systems:Metabolism anomalies:Hyperglycemia/diabetes mellitus

1 to 7 out of 7

- 3199 Stimmler syndrome
- 2579 Muscular atrophy ataxia retinitis pigmentosa diabetes mellitus
Furukawa takagi nakao syndrome
Cerebellar ataxia, autosomal dominant
- 99 ADCA
Spinocerebellar ataxia, autosomal dominant
- 100 Ataxia telangiectasia
Louis-Bar syndrome
- 3349 Treft-Sanborn-Carey syndrome
Optic atrophy - ophthalmoplegia - ptosis - deafness - myopathy
- 2047 Flynn aird syndrome
- 2377 Laurence-Moon syndrome

Research project by Dynamic Taxonomies Research Group | Rare disease database by Orphanet | UKP software donated by Knowledge Processors US patents 6,763,349 and 7,340,451, pat. pend.

Fig. 9.10C Reduced taxonomy and diseases after zooming on *ataxia* and *hyperglycemia*

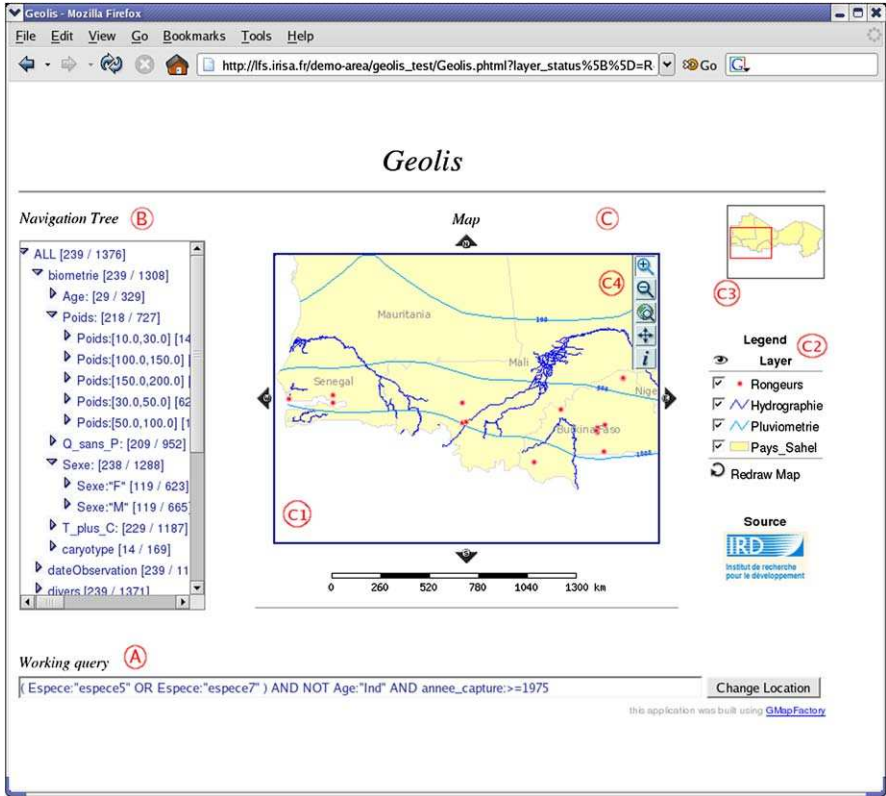


Fig. 9.13C The web interface of GEOLIS