INFORMATION RETRIEVAL AND ACCESS

On-line documents include electronic mail, technical papers, newswire articles, inter-office memos, company policies, advertisements, and countless other styles of information. There are now so many documents stored electronically that it presents a challenging problem for anyone who needs to find useful information within them. A library catalogues all of its books and assigns them subject headings or classification codes. On-line documents sometimes have such retrieval aids, but generally do not: other than the To, From, Subject, and Date fields, an inter-office memo rarely contains clues about its content. With or without manually added subject headings, effectively building and using automatic indexes for online documents is a difficult task. The research field of information retrieval (IR) has been addressing the issues that have arisen since the late 1950s.

Although it is actually broader in scope, information retrieval is best understood in the context of documents and user queries. In that setting, the "information" is a collection of electronic textual documents that might cover a narrow subject area, but more often ranges over a wide array of topics. A user is interested in answering a question and believes that the answer may lie somewhere within the collection of documents. Information retrieval, then, is the process of acquiring a written query from the user, matching it against the documents in the collection, and presenting those documents in such a way that the user can readily find the ones most likely to be useful.

A traditional database search is a special case of information retrieval. In relational databases, the user's query specifies names of fields and what their contents must look like—for example, "get employees with name field containing SMITH,J and salary over 30,000" (which would actually be expressed in a query language such as structured query language [SQL]). The fields have semantics, and the user knows

what their meanings are: the name field is of "surname, given name" format, and the salary is in U.S. dollars. The field of IR expands the capabilities of databases substantially by allowing searching of unformatted text fields with unspecified content. A database system might allow a search for "name contains SMITH," but IR allows a search for documents that discuss "the economic impact of recycling tires." A good IR system might not even require that any of those words appear in the document: a text discussing the creation of jobs by a rubber re-processing factory might be retrieved!

IR should not be confused with text understanding or Natural Language Processing research. IR takes advantage of generalizable results in those areas, but is concerned with locating useful text *without* deep understanding. This choice is not because deep understanding is bad, but because it is currently unattainable in any general setting. IR therefore focuses on low-level statistical properties of words, phrases, etc., and relates them to meaning to whatever extent possible.

Overview of Information Retrieval

The motivation behind most research in IR is to make it easy for users to find documents that might answer their questions in a short time. This could involve improving the techniques for comparing a user's query to the documents in the collection, summarizing a document for quick perusal, adding a better user interface to a search system, or an amalgam of many such techniques. Although some IR systems and research use a restricted query language (e.g., requiring Boolean operators), the bulk of the effort has focused on systems that accept free-text queries from a user and match them against the document texts. Research has considered problems during the entire process: from evoking a "good" query from a user, through analyzing the query to help match it against the corpus, into the process of matching queries and documents, the ranking of matched documents, through presenting documents, and even feeding back information to begin a new cycle of retrieval.

IR has many sub-areas of research with different relative importance of those problems:

- In "ad hoc" retrieval, the most commonly encountered style of IR, the system processes a new query against a known collection—the user runs a query against a collection that is likely to contain the answer. For this type of IR, evoking useful queries from the user is extremely important. Matching the query against the document collection is motivated by striving for high-quality results in the top 10–20 documents selected with less concern for the lower ranks.
- The "filtering" application, on the other hand, matches a stream of incoming new documents against a set of long-lived queries, providing targeted "news clipping" services automatically. A good starting query is useful in this setting, but interaction with the user over time will allow the system to customize the query very closely to the user's needs. An important problem for this application is identifying borderline documents that are harder to separate between useful and useless.
- Summarization research investigates how the retrieved documents can be presented more concisely, either by collapsing a document to its few key sentences or by extracting themes that span the entire set of retrieved documents. The quality of a query is clearly not important here (unless the summaries are query-centered), but methods of text analysis like those used to parse queries are useful.

The following sections discuss each of those basic areas as well as the methodological and theoretical underpinnings of IR systems and research. In addition, this article covers interactive IR, user interfaces, and visualization techniques, all of which have been elevated in importance in the recent growth of generally available computing. The last section of the article presents the relatively new sub-field of multimedia IR: applying the ideas behind IR's methods to non textual sources.

INFORMATION RETRIEVAL AND THE INTERNET

With the advent of the World Wide Web ("the Web") the amount of textual information available electronically has increased by several orders of magnitude. Along with this increase in the available text, the problem of finding desired information has also grown substantially. Web search engines, based on information retrieval technology, offer an effective solution to this problem. There are two essential components that make a Web search engine work, one that gathers the information available on the Web into a large central repository, and another that creates indices on this repository and allows people to search through the index using IR techniques.

The information gathering component needed to populate the index of a Web search engine is often called a Web robot. It is a computer program that "crawls" from document to document on the Web, traveling along the links between the hypertext pages it encounters. This style of gathering documents also results in the programs being called Web Spiders or Web Crawlers (1). The pages gathered by a robot are passed to the main information retrieval component of a Web search engine that allows people to search for useful pages. Web search engines are usually based on (or similar in design to) one of the popular IR models discussed below.

Search systems consist of indexing and retrieval components. The engine first "indexes" a collection of documents—in the case of the Web search engines, the collection was gathered by a Web robot; in other settings, the documents would have been gathered differently. At a high level, indexing includes the following steps:

1. A page is first "tokenized" to obtain individual tokens. In this process, each word on the page becomes a token. Formatting markup (e.g., HTML tags) is not considered part of the document and is rarely tokenized.
2. Stop words are removed. Function words like *the*, *in*, *of*, *an*, etc. are removed, as these words do not convey any content of what the text is about. They are called "stop words" because their indexing is "stopped" at this point.
3. Words are often reduced to their morphological stems; for example, the words running, runner, and runs might all be reduced to a single stem run. This stemming process varies from system to system. Some systems do not provide automatic stemming, instead providing trunca-

tion options in the query language. For example, using "run*" means to match all words starting with "run"—an awkward approach since it would also match runt and rung.

4. Some systems provide an optional step of discovering phrases in the text and adding them to the list of "words" as additional index terms for the page. A system might recognize proper names (e.g., United States of America) or more general phrases (e.g., political campaign). The types of phrases and methods for finding them vary widely, from elaborate syntactic processing of language to purely statistical approaches.
5. Optionally, a thesaurus is used to add all possible synonyms of the words on a page to the list of index terms for the page. Most systems defer thesaurus use to retrieval since a thesaurus is very difficult to apply automatically; words have too many different meanings.

Given a word or phrase, an inverted index points to all the documents that contain it; an entry in this index is stored for every term encountered in the collection of documents (other than stopwords). An inverted index is the most popular method for storing an IR index because it is highly efficient for a sparse index (most words occur in a small fraction of the total number of documents).

In the search phase, a user enters a query, and the IR system uses the inverted index to find all pages that contain the query word. It is straightforward to combine two inverted lists to find documents that contain either word, that contain both words, or one word but not the other. The system's query language allows a user to select various combination options along those lines. These documents or pages are then ranked in the order of their perceived "goodness" for the user query and are presented to the user in that order. Classical IR models are used to generate a rank ordering for the documents as discussed below in the section on IR models.

EXPERIMENTAL EVALUATION

Experimental evaluation has been a cornerstone of information retrieval research throughout the entire history of the field. When dealing with something as ambiguous and complex as language, there is very little that can be "proven" about it (other than the fact that language processing is a complicated task); one can only "show" that certain techniques or models work better or worse than others. To show the relative merits of different techniques, an objective evaluation energy—one that measures the success of a technique—is needed. Most research in information retrieval is conducted by

1. developing a hypothesis for how a particular task can benefit from a certain technique
2. implementing that technique in an automatic system and experimenting with the selected task and
3. evaluating the results

For example, the ad hoc task (in which documents are ranked for a user's new query) can be evaluated by measuring how many "good documents" were ranked high and were easily accessible to the user. A system that ranks many good documents higher than most bad documents is certainly more effective than another system that doesn't. To do such an evaluation, one needs a set of documents (the corpus), a set of user queries (the query set), and a definition of "good documents" and "bad documents" for each query (the relevance judgments). Given this test collection, multiple techniques can be used to rank the documents in the corpus for various queries, and using some measure of how good the rankings are, one can compare the ranking effectiveness of various techniques.

The idea is that automatic systems will simulate having a user run their system. Two approaches that are being compared start with the same query from the test set. They each run the query using their own purely automatic techniques to get a list of documents ranked in the order they expect will most likely reflect the user's notion of relevance. The two rankings can then be compared because the test collection includes a list of all of the documents that were, in fact, relevant to the user. The experiments can be run repeatedly and evaluated objectively. Although the judgments themselves came from a person and so are subjective, once fixed, their use for evaluation is objective.

Test Collections

Constructing a test collection is an expensive task. Gathering a large collection of on-line documents is no longer a large problem (except, perhaps, for acquiring permission to use the material). The difficulty is the time and money needed to involve humans in the process. Not only do the users have to provide a set of "real" queries, but they need to read large numbers of articles and mark which articles were relevant to their query and which were not. For a large test collection, that could require reading thousands of articles over several weeks. Involving many users for the large amount of time needed to make the relevance judgments is quite expensive but is of tremendous value for researchers trying to objectively evaluate their techniques. In recent years, the U.S. government, under the sponsorship of NIST (National Institute of Standards and Technology) and DARPA (Defense Advance Research Project Agency), has been running a program called TREC (Text REtrieval Conference) to evaluate various techniques for text processing objectively (2). A critical by-product of TREC is the creation of a large test collection for IR system evaluation. In TREC, researchers are provided with a large set of documents (several gigabytes of text) and a set of user queries. Researchers and IR companies use their systems (without knowing the relevance of articles) to rank the documents in the collection for the queries. After all systems have submitted their results, the documents in the ranked list are evaluated by the users for relevance. The resulting judgments serve as a blind test of performance of various systems and techniques, and also create an extensive set of relevance judgments for the user queries, yielding an extremely valuable test collection for further information retrieval research. From the Cranfield collection in the 1960s to the TREC collections in the 1990s, test collections have served as the foundation for most IR research.

Evaluation Measures

To evaluate a system, the goodness of a ranking is usually measured in terms of recall and precision. If a certain set of documents is retrieved for a query, then recall is the propor-

tion of the relevant documents that are retrieved (the coverage of the retrieved set); whereas precision is the proportion of the retrieved documents that are relevant (the accuracy of the retrieved set). The main assumption behind the use of measures such as recall and precision is that a typical user wants to retrieve a high number of relevant articles (achieving high recall) and does not want to retrieve too many nonrelevant documents (maintaining high precision). Traditionally, these two goals have proven to be contradictory: as an extreme, a system can achieve perfect recall by returning the entire collection to the user, resulting in quite poor precision. Since a ranked system usually does not retrieve a fixed set of documents, but instead ranks all the documents in the collection, researchers use average precision to evaluate their systems, extending the notion of recall and precision to ranked retrieval. The main idea behind average precision is to move down the document ranking and repeatedly compute the precision after a certain number of new relevant documents have been seen. For example, calculate the precision after the first relevant document is found, after the fifth, and so on. To some extent, this approach counts how many nonrelevant documents a user must see in order to find the desired number of relevant documents. Finally, all those precision values are averaged to get the overall system performance. Precision can also be measured after a fraction of the relevant documents have been found (rather than a fixed number). For example, it can be calculated after 10% of the relevant documents have been seen, after 20% of the relevant documents have been seen, and so on up until all (100%) have been seen. These precision values at the various recall points (10%, 20%, . . ., 100%) are then averaged to compute the average precision of a system. A system might perform very well on a handful of queries but fail miserably on numerous other queries. To compensate for such query-specific biases, results for multiple queries are further averaged to obtain the overall system score. The intent is to measure the average effectiveness of a system over many queries—depending how the test collection was built, that is similar to evaluating it over many years. See (3) for a more detailed discussion of evaluation.

It is difficult to emphasize enough the usefulness of the test collections for the development of better search technology. To give the reader an idea of how valuable the TREC collection has been to experimental IR, since the inception of TREC in 1992, average precision for retrieval systems has more than doubled in five years. What this means to a user is that if the top ten documents contained on average four relevant documents in 1992, a current, state-of-the-art system is likely to retrieve eight in the top ten. This dramatic improvement is the direct result of researchers' testing their ideas on these test collections and developing new and effective techniques.

MODELS FOR INFORMATION RETRIEVAL

A model is a formalism that is used to translate real life problems in IR into a domain, usually mathematical, in which one can argue methodologically about the problems. For instance, in the vector space model, each text is translated into a vector; formal aspects of vectors can then be applied to texts. Use of formal models allows researchers to define mathematical measures of text relatedness, that in turn, are used in various IR tasks. For example, for the ad hoc task, documents can be ranked in decreasing order of their relatedness to a user query. A formal model also helps researchers when analyzing the strengths and shortcomings of various techniques. More importantly, it provides them with a tool to visualize or reason about a problem that often leads to natural and effective solutions to the problem. The experimental evaluation methods described above allow a researcher to test the validity of a model: does it accurately describe what happens, and does it allow the researcher to predict new results?

The following sections briefly outline four popular models of IR: Boolean, Vector space, Probabilistic, and Network. There are a large number of variants on these four, and there are also other models totally unlike them (e.g., terminological, topological, etc.). However, the four below are well-known and the basis for both research and commercial systems.

Boolean Model

The Boolean model was primarily designed for the ad hoc task. In this model, a query is a Boolean statement involving words and phrases. For example, the query "(information OR text) AND retrieval" might be used to find any document that has the word "retrieval" but that also must have either the word "information" or the word "text" (or both of them). All documents that satisfy a query are retrieved. The main problems with this model are (a) it does not rank documents, and (b) users find it difficult to form good Boolean queries for nontrivial concepts. Document ranking can be achieved by using a soft-Boolean model (e.g., fuzzy matching) that incorporates the notion of a partial match for Boolean queries, but the difficulty of forming complex Boolean queries still hinders widespread effective use of this model. The Boolean model is very commonly used in library catalogue systems where the controlled vocabulary of subject headings makes it possible to construct Boolean queries with reasonable accuracy.

Vector Space Model

In the vector space model, a high dimensional vector space is constructed such that each word in the vocabulary is an independent dimension in this space. For a large document collection, there might be a million words in the vocabulary—the vector space model in that case manipulates vectors of one million dimensions.(4)

Each text (a query, a document, a paragraph, etc.) is converted into a vector in this high-dimension space. As a synthetic example, consider what happens if there are only two words in our vocabulary: information and retrieval. Any utterance from this vocabulary can be mapped to a two-dimensional vector. If the X-dimension corresponds to the word information and the Y-dimension to retrieval, then the text "information retrieval information" can be a vector ($X = 2$, $Y = 1$) in this space. The magnitude of a vector in any particular dimension corresponds to the importance of that word in the text. Term weighting schemes (discussed below) are used to assign that magnitude.

All vector operations can now be applied to texts. To measure the semantic-relatedness of two texts, the model states that one compute the distance between the corresponding vectors. If the vectors are far apart in the vector space, then the two texts are not well related, but if the vectors are close to each other, then the two texts are strongly related. In the ad

hoc task, such measures are used to find the distance of a query vector to all the document vectors. The documents are ranked in increasing order of their distance (decreasing relatedness) to the query.

Words within a vector are assigned numeric "weights" that reflect their importance in the text. A frequently used word is usually more important in a text, whereas a word that is widely used across documents is usually a common word and is often not as important. In particular, such frequently-used words have little value in distinguishing between texts. For example, in a collection of computing reports, the word "computer" probably occurs in every document, and is, therefore, useless as a retrieval clue. In a general collection of texts, the word might occur in 1% of the documents, and would, therefore, be more useful for pulling out the documents about computing.

The importance of a term within a document is reflected by counting the number of occurrences of a word in a text and incorporating that into the weight of the word. The IR research community refers to this value as the term frequency (or tf). How widespread the word is used is measured by an inverse function of the number of documents containing the word. If more documents use a word, it is considered less important; the fewer documents that use it, the more valuable it is. Typically, a function is used similar to $\log(N/n)$, where N is the total number of documents in the collection, and n is the number of documents that contain a word. That value is called the inverse document frequency or the idf-factor. Using these two factors, the weight of a word in a text becomes: $tf \times \log(N/n)$. The combination is referred to as "tf-idf weighting" and is the basis of almost all popular IR systems' weighting schemes (5).

When a user poses a search query, the query is also converted into a weighted vector. A vector similarity between the query vector and every document vector is computed, and documents are ranked in decreasing order of their similarity to the query. Typically, the vector similarity is computed as the inner product between the query vector and the document vector; that is,

$$\mathrm{Sim}(Q, D) = \sum_{t_i \in Q} q_i \times d_i \tag{1}$$

where q_i represents the weight of a query term t_i, and d_i is the weight of that same query term in the document D. When the document and query vectors are normalized to have unit length, the similarity is exactly the cosine of the angle between the vectors. For that reason, this measure is often referred to as "cosine similarity."

The Smart information retrieval system is one of the most popular implementations of the vector space model. Developed in the 1960's, and enhanced extensively since then, the Smart system has been the source of many advances in the field of IR. Its main objective is to serve as an experimental IR system in which people can implement their own ideas rapidly and test them without having to implement a complete search engine. Because of its flexibility and availability, the Smart system has been one of the main engines used by researchers over the years.

Probabilistic Model

This model is based on probability theory. A key hypothesis of the model is the Probability Ranking Principle: if a system ranks documents in the order of probability of relevance to the user who submitted the query, then the effectiveness of the system will be optimal given the information available to the system. That is, documents are ideally ranked in the order of P(relevance|document, query). Such a ranking is claimed to be optimal, and additionally, the user is presented with a probability that the document will be useful.

Most uses of the probabilistic model appear to Bayes' rule to help implement a system that can estimate the probabilities. Specifically (assuming a fixed query from here on),

$$P(\text{relevance}|\text{document}) = \frac{P(\text{document}|\text{relevance}) \times P(\text{relevance})}{P(\text{document})} \tag{2}$$

For reasons of efficiency and to eliminate the need for estimating the prior probabilities of a document and relevance, this can be converted to the odds of relevance:

$$\frac{P(\text{relevance}|\text{document})}{P(\text{nonrelevance}|\text{document})} \tag{3}$$

The document ranking is the same (so the Probability Ranking Principle is satisfied), but the resulting score is no longer the probability of relevance. More importantly, the expression can be simplified by applying Bayes' rule and recognizing that all the prior probabilities are constants for a particular query. Documents can then be ranked by the simple ratio

$$\frac{P(\text{document}|\text{relevance})}{P(\text{document}|\text{nonrelevance})} \tag{4}$$

Assuming that words' occurrences are independent of each other, Eq. (4) can be transformed into the following expression:

$$\frac{P(\text{word}_1|\text{relevance}) \times P(\text{word}_2|\text{relevance}) \times \ldots}{P(\text{word}_1|\text{nonrelevance}) \times P(\text{word}_2|\text{nonrelevance}) \times \ldots} \tag{5}$$

where each word is a query word. Using log odds rather than just odds yields

$$\sum_{w_i \in Q} \log P(w_i|\text{relevance}) - \log P(w_i|\text{nonrelevance}) \tag{6}$$

Under the reasonable assumption that most of the documents in a collection are nonrelevant, if a collection is large enough, the probability that a word occurs given relevance is close to a constant. Also, the probability of a word given nonrelevance can be estimated by the fraction of the documents that contain the word (i.e., n/N, where N is the total number of documents in the collection, and n is the number of documents that contain the word). So the second factor changes to $-\log(n/N)$ or $\log(N/n)$. Those assumptions result in simplification of the log odds ranking function to

$$\sum_{w_i \in Q} \log(N/n_i) \tag{7}$$

Observe how this formulation is close to the vector space ranking function of the documents if all query words are assigned a unit weight, and the term frequency factor is ig-

nored. The probabilistic model provides a theoretical justification for the "idf" component of term weighting that was developed empirically in the context of the vector space model. See Ref. (6) for the details of this derivation: Ref. (3) provides a nice presentation of probabilistic IR.

Very few major IR systems are based purely on the probabilistic model. The Okapi system from City University of London (7) is a well-known research engine based upon probabilistic assumptions.

Network Model

A Bayesian belief inference network is a mechanism for combinations of evidence that lead to a belief that something is true. In the context of IR, that means the belief that a document is relevant to a query. The power of developing an IR model from the inference network is that it allows the model to incorporate evidence from different sources easily and combine them in a theoretically justifiable way. For example, an inference-based IR system might find evidence for relevance based on the strength of query words occurring in the document, but could combine that with evidence from phrases that include the word or from manually assigned keywords. The system could also take evidence from alternative representations of the same document, increasing its confidence if a query matches the document as well as a document summary that happens to be a second document. Any other model can do the same, but the inference network provides a means to do so that is modeled and can, therefore, be tested and used to make predictions.

In practice, the belief networks need to be slightly simplified for efficient implementation—for example, limiting the types of evidence combinations to those that can be calculated quickly, and removing any chance for cycles in the network. The InQuery search engine (8) developed at the University of Massachusetts is a popular research and commercial IR engine that is based on the inference network model.

AD-HOC RETRIEVAL

One of the fundamental tasks of information retrieval is searching an existing collection of documents in response to a user's query. When a user has a new question and creates a query for that question and does not save it longer than the few minutes needed to use the IR system, the process is known as "ad-hoc retrieval." The name is meant to distinguish it from other tasks where the query is long-lived.

An ad-hoc query is usually constructed by a user who has a sense of the words that will be used in a relevant document. The IR system's query language provides a variety of methods for specifying how those words are likely to be related. Exactly which operations are available usually depends on the theoretical model underlying the system. For example, operators like the following might be used to relate two words.

star bright indicates documents containing either word or both of them, in any order, with neither word being more important. This sort of query is particularly common with a vector space model where word order is generally not considered, but virtually every IR system supports something akin to it.

"star bright" selects documents that contain the phrase "star bright"—that is, the two words immediately adjacent and in that order. Some systems might relax the phrase operator, allowing a match if the words appear within a few words of each other. Other systems use a special operator—e.g., #phrase(star bright)—to select such a relaxed view.

3.0 star 20.4 bright is similar to the first example but puts a weight on the two words, indicating that "bright" is almost 7 times as important as "star." This sort of option allows relative importance of terms to be specified. The weights correspond to the values q_i referred to in the vector space discussion above.

star BUTNOT bright specifies that documents containing "star" should be retrieved, provided that the word "bright" does not also occur in the document. This style of query is generally seen in a Boolean system, but variants are not uncommon in probabilistic systems. Probabilistic systems generally support "looser" versions of the Boolean operators that produce probabilities or numeric beliefs rather than strictly TRUE and FALSE values.

An extremely large class of IR systems expect the user to enter a "natural language" (free text) query and then attempt to transform it into the systems internal query language automatically. The TREC evaluation's "ad hoc" task follows this approach, providing a full-text description of a query (not in any system's query language) and requiring that participating systems automatically convert the description into a query. TREC also provides a "manual" version of the ad-hoc task that allows an IR system to demonstrate its power if it is used in the way the system was designed.

Stopping and Stemming

Most IR systems provide behind-the-scenes stopping and stemming methods in addition to the query operators. Stopwords are words that are not content bearing: they do not in and of themselves provide any clue as to what a document is about. For example, the is a stopword because it is a determiner with no real meaning itself, and heretofore might also be a stopword because although it seems more useful than the does, it still is not very descriptive of the document. (These words may all be used in combinations with other words to derive features more complex than single words, of course. In "Winnie the Pooh," the middle word is useful even if it still provides little content.)

Stemming is the process of stripping suffixes (and prefixes) from word forms so that morphologically related words are grouped together. For example, the words worker, working, and works are all variants of work, and a user searching for one probably is interested in any of them. Stemming is complicated by two issues. First, it is difficult to build a general-purpose, rule-based stemmer that does not make mistakes—for example, conflating police and policy because *ice* and *icy* are related—although some dictionary-based stemmers prevent the most egregious of those errors. A second issue that arises in stemming is that although it generally makes sense to conflate two word forms, one of the forms has an alternate meaning that should not be conflated. For example, the word gravity can refer to serious (grave) situations and so should be stemmed to grave, but both gravity and grave have alternate meanings (the Earth's pull and a cemetery plot) that will decrease retrieval effectiveness if conflated. Stemmers that take advantage of part-of-speech taggers can solve some of these problems, but there are countless exceptions that are

difficult to address. Evaluations of stemming efforts have shown little difference between various types of stemming: each stemmer makes its own characteristic mistakes, so on average, they perform very similarly.

Query Refinement

The disadvantage of automatic query formulation is that mistakes in the processing can result in a weak or even incorrect query. In the latter case, there is little to be done but to reformulate the query so the system can do better (see below for a discussion of interactive techniques that address this problem). However, if the generated query is weak—retrieves only one or two marginally relevant documents—it can often be refined by a process known as "relevance feedback."

For relevance feedback to be useful, the user needs to have seen a couple of documents that are deemed "on topic" to the information need. The system, when presented with a few sample "good" documents (and optionally some explicitly marked "bad" documents), can analyze the documents for patterns that reflect relevance and use them to modify the user's query. Patterns are most commonly words, phrases, pairs of words that occur near each other, and so on. Because the patterns that relevance feedback finds are derived from the documents in the collection, there is the expectation that the patterns will apply to remaining documents in the collection—preferably to the relevant documents.

Relevance feedback can be used to adjust the weights on a user's query terms automatically, but is most effectively used to expand the user's query by adding new terms, phrases, or other features that help recall relevant documents and also distinguish them from nonrelevant texts. Relevance feedback is a highly effective technique, improving the quality of retrieval 25–75%, depending on the query and the collection.

Relevance feedback can also be used automatically, in a process known as "local feedback" or "pseudo-relevance feedback." To operate automatically, a system retrieves documents in response to a query but does not display them to the user. Instead, the system behaves as if the user had marked the most highly ranked documents as relevant and modifies the query. The new query is run and only then are the retrieved documents displayed.

This automatic technique is highly effective for large numbers of queries, though the quality of its result depends greatly on the quality of the initially retrieved documents. If the highly ranked documents chanced all to be nonrelevant, the query will be automatically adjusted to retrieve nonuseful material. On average, however, this technique is helpful and yields a 10–20% improvement in average precision.

Cross-Language Retrieval

Variants of ad-hoc retrieval are appearing as the number and sources of on-line documents increase. An intriguing example is that of cross-language retrieval. In this ad-hoc setting, the user poses a query in one language in order to retrieve documents in any other language, presumably only those in which the user has a reading knowledge—allowing multilingual users to avoid typing the query many times, and also serving people who have a reading knowledge in a second language but cannot adequately form a query in that language. These techniques tend to be based upon statistical relationships between words in the languages rather than on elaborate Machine Translation methods.

INFORMATION FILTERING AND TEXT CLASSIFICATION

Another very useful facet of IR is information filtering. With the amount of electronic information that is generated everyday, it is hard for a person to get interesting information without weeding through a lot of uninteresting information. Information filtering aims to select from a continuous stream of articles only those that are potentially interesting to the user (9). An example of information filtering would be a personalized news clipping service. Users communicate their interests to a filtering system, typically in a simple natural language statement. The system matches all new articles with a user's interest statement, and if an appropriate match exists, an article is sent to the user. Over a period of time, a user might indicate to the system which of the articles that the user received were actually useful, and based upon that, the system can "learn" the user's preferences and modify the user's profile to do better filtering in the future.

Most current information filtering systems convert a user's statement of interest into a "user profile." Typically, a user profile is a weighted list of words and phrases (as in the vector space model). The higher the weight of a word or a phrase in a user profile, the greater are the chances that if that word appears in a new article, then that article will be interesting to the user. Initially, when the user has not indicated to the system what articles he or she liked, words in the user's interest statement are used to create the user profile. But once the user has provided preferences to the system, the system can use this information to build a better user profile using the query reformulation techniques discussed in the context of ad hoc retrieval above (e.g., relevance feedback). To learn the words and their weights for a user profile, most filtering systems use the probability of occurrence (or some variation of it) of a word or a phrase in the articles marked useful by a user and in the nonrelevant articles. A word that occurs with very high probability in the articles marked relevant by a user but that occurs with low chances in the nonrelevant articles gets a high weight (high importance) in the user profile, and vice-versa.

Using the user profile as a long-standing query, every new document is matched to this query using standard IR matching techniques. If a document exhibits a good match to a user profile, the document is sent to the user, otherwise not. Typically, a numeric score is computed for a document with respect to a user profile. If this numeric score is greater than a certain "goodness" threshold, the document is assumed to have a good match to a user profile and is sent to the user. Many commercial organizations run a news filtering service. They buy information, typically news stories, from information providers like new agencies, publishers, and other information sources. Each story in this electronic newswire is matched against a large number of customer profiles, and the appropriate stories are sent to the customers who might be interested in them.

Text Classification

A very related application of IR technology is text classification. In many circumstances, articles must be assigned to one

or more of a predefined set of categories—for example, assigning news stories to particular sections in a newspaper, or given a patient's health summary, assigning a diagnosis code to the patient's record for billing purposes. Many organizations have their own coding techniques to classify information into many possible codes or classes. Text classification is very similar to the information filtering task. In this task, there are categories (or classes) instead of users, and category profiles are built instead of user profiles. There is no initial statement of interest from a user; instead, there are some preclassified examples for each category that are used to learn the category profiles. A profile for a class is built using techniques very similar to those that are used to build a user profile in information filtering (after the user has provided a system with some feedback of goodness of documents). To classify a document, the score for the document is computed for every category, and the document is assigned to the class or classes with which it has a good match.

In the absence of an automatic system, the task of text categorization is often done by subject experts—for example, newspaper editors decide which story should be published under what heading. But as the amount of text to classify increases, and the number of categories increases, it becomes hard for a human to remember all the possible categories for an article—or to process the entire volume of data that arrives. Text categorization has been successfully used as an aide to a human expert; instead of trying to assess the possible classes for an article from scratch, the subject expert asks a text categorization system to "suggest" a few classes in which the document can be placed. It is then much easier for the human expert to decide if indeed the article belongs to some of those classes or not. Large reductions in human classification time have been reported by the use of a classification system as an initial class proposer.

DOCUMENT SUMMARIZATION

As the amount of textual information available electronically grows rapidly, it becomes more difficult for a user to cope with all the text that is potentially of interest. For this reason, automatic document summarization is an important part of text processing. Unfortunately, this area has also proven to be one of the more elusive tasks of the field; over the last forty years, researchers have tried and developed numerous techniques for text summarization, but they have been unable to develop a general purpose, domain independent, text summarizer. Even though automatic text summarization techniques have been moderately successful in very narrow domains, most current domain independent summarization techniques are mediocre at best (10,11).

For documents in narrow domains with predictable characteristics, a detailed semantic analysis can be performed to construct an abstract representation of the meaning of a text. Such an analysis typically yields a set of frames that have been manually tailored for the particular domain. Domain dependent text generation techniques can then be used to generate fluent summaries for documents. Such systems have been developed for documents on corporate mergers and acquisitions, texts on micro-processor systems, stories in terrorism domain, diagnostic messages in automotive equipment failure, reports of baseball games, and several other restricted domains. Unfortunately, such techniques depend on the presence of large and complicated knowledge bases for every domain. Building such a knowledge base requires intensive manual effort for a given domain, prohibitively expensive for more than a handful of subject areas. For that reason, domain dependent summarization techniques do not scale up to yield a general purpose text summarizer.

For unrestricted domains, automatic text summarization is mostly done by text extraction. Pieces (sentences, paragraphs, etc.) of a given document are deemed important—based on some statistical characteristics of a piece—for inclusion in a summary. The most important pieces are then extracted and concatenated together in the order they appear in the original text to obtain an extract, or a representative summary, of desired length. Sentence extraction is most often used for summarization by text extraction. (12). In sentence extraction, clues to the usefulness of each sentence are used to score sentences of a document, and the most important sentences are extracted. Typical clues that indicate the usefulness of a sentence are

- The sentence contains important keywords. Keyword frequencies and tf-idf weights in the document are used to determine the importance of various keywords.
- The location of a sentence is well known to be useful. The first and sometimes the last sentence of a paragraph tend to be more useful, and paragraphs (and their sentences) near the beginning and the end of a document contain more important material.
- A sentence contains word cues. Sentences containing superlatives and value words like greatest and significant tend to be important.
- The sentence contains indicative phrases. Sentences that contain phrases like "the purpose of this article is to . . ." or "our investigation has shown that . . ." are typically useful.

A good summary has two desirable properties: coverage and cohesion. Coverage refers to the amount of information from the original text that is presented by a summary. Cohesion is aimed at the readability of a summary. Sentence-based summaries can achieve reasonable coverage, but since isolated sentences from different paragraphs appear next to each other in the summary, the cohesion of such summaries is usually poor. Researchers have also extracted text pieces bigger than a sentence in summarization by extraction. Since bigger text pieces also include some context in which sentences occur, it is expected they will yield more readable summaries. But extracting larger pieces, like paragraphs, comes at the cost of poorer coverage given a fixed size of the summary.

Researchers have also explored middle ground that aims for domain independent sentence extraction of articles in a single genre—for example, technical papers. By hand tailoring the sentence extraction rules for articles of a single genre, they expect to get more useful summaries than the summaries obtained by a truly unrestricted (genre- and domain-independent) sentence extractor. In reality, true domain independence is not achieved, and some tuning of rules is also needed for each new domain. For example, a rule for technical papers might be: If a sentence begins "This paper studies . . ." then extract the sentence. Such hand tailored rules

along with sentence extraction provide summaries that are possibly more readable and informative than simple sentence extraction summaries, but they come at a cost of manual creation of rules for every new genre and domain.

INTERACTIVE SYSTEMS

Automatic IR techniques are quite good, but the state of the art still leaves room for plenty of mistakes. Because IR systems are often used interactively, there has been substantial research into how an IR system can present information in a way that lets the user guide the system or helps the user figure out rapidly when the system is going astray.

Interactive IR work can be divided into three camps. First, there are attempts to develop better interfaces for users, sometimes general purpose interfaces, but often more task-specific interfaces. Second, some types of information-seeking behavior are better characterized as browsing than as searching, resulting in interactive browsers. Finally, another important area of interactive IR is visualization of retrieved information in an effort to help the user understand what the system did or how the system might be corrected.

Information Retrieval Interfaces

Interface development in IR is usually informed by specific tasks related to searching for information. A high-quality interface needs to be designed with the basic process of IR in mind. One way of meeting those needs is indicated by a several-faceted approach to IR systems (13). Each "facet" represents an important step in the process of forming and running a query:

1. Query formulation. This step involves selection of sources, fields that will be searched, how the query will be represented, and variations in terms that should be allowed or required.
2. Beginning the search. Most IR systems start the system with a "go" action of some sort, but there are some systems that are continually modifying the output as the user enters information—for example, systems that display all possible completions of the current keyword as it is typed.
3. Review of search results. This step includes presentation of ranked lists, browsing capabilities, and visualizations.
4. Refinement. At this point, the user may refine the query using techniques such as Relevance Feedback, and then resubmit the query, starting over from step (1).

The intent is that if an interface designer keeps all of these steps in mind, the interface will be intelligible to novice users and useful to experienced users. Any interface that obscures one of the facets is likely to be less usable. All interface work is heavily tied to the research work in the human–computer interaction community.

Browsers

The query formulation phase of an IR system is sometimes difficult because the user does not know enough about the collection to form a query that will match the content of relevant documents. Interactive browsing tools are an ideal means of addressing that confusion. Browsing tools are commonly used to reveal the possible query words to the user—for example, systems that show all possible words starting with the first few letters a user has typed. Collections that include database-like field information (e.g., author name or creation date) are more useful when those fields can be browsed—for example, for names starting with a particular sequence or names sounding like a phonetic string.

More interesting browsers, however, expose more information about the collection and the relationships between its concepts or its documents. For example, IR researchers have investigated the effect of "clustering" the documents in the collection to help the user understand how the documents clump together into groups (when they do). Clustering works by locating sets of very similar documents and grouping them together. The process continues by looking for other document pairs that are similar, and by expanding to consider clusters that are similar, until some appropriate number of clusters has been found. ("Similar" is defined by the theoretical model underlying the system, but is most often based on the vector space model.)

A recent system that demonstrates these ideas was developed at Zerox's Palo Alto Research Center in the early 1990s (14). This system, called Scatter/Gather, is based on a very fast algorithm that allows even fairly large collections to be clustered into five or so clusters rapidly. A user can then choose one or two of those clusters and have the documents in those groups quickly reclustered. This allows the user to "dive down into" a collection and understand what is in it. One unfortunate aspect of clustering a large number of documents into a small number of clusters is that groups are not always homogenous, resulting in disconcerting relationships at times (e.g., a cluster apparently about mid-East peace process including documents discussing Alaskan fishing).

A modification to Scatter/Gather addresses the problem of nonhomogeneity by clustering only the documents retrieved in response to the query rather than the entire collection. The set is smaller and more directed, so the clusters are more likely to be tightly focused. An additional benefit of using the retrieved set only is that it has been known since the 1970s that relevant documents tend to cluster together—this browsing method, therefore, not only lets the user see how the retrieved documents group together, but may also enable a user to find the relevant documents more quickly.

Visualization

Within IR, visualization provides means for looking at the contents of a document collection—or at the results of a search—in some way other than examining the text. An extremely simple visualization is one that replaces a retrieval score with a histogram. It is often easier for a person to see the relative differences between bars on a histogram than between numbers, so this visualization helps the user more rapidly understand the search results.

Other visualizations that aid comprehension include showing how the words in the query occur within the document; the portions of the document that are strongly related to the query are evidenced by a higher concentration of matches. Visuals that cluster related documents together (as in Scatter/Gather, above) and show the documents as nodes and their

relationships as edges in the graph can expose the possibility that there are multiple types of retrieved documents—for example, because a query word has multiple senses (river bank and money bank). Scatter/Gather itself shows the clusters as groups of significant keywords and titles in the cluster rather than a graphical representation.

Visuals that show nodes and relationships between them are also useful for browsing. The documents in a collection (or a retrieved set) can be represented as stars in a "universe," where strongly similar documents group together to form galaxy-like clumps. A user can browse through the visualization as if flying a spaceship through the universe.

Another approach to browsing shows not the documents but the concepts (words, phrases, etc.) in a collection and how they relate to each other; for example, they are related because they co-occur in many documents. The concepts might be shown in the same star-like display, with the "galaxies" now representing groups of strongly-related terms. The concept display could be combined with the document display to show the relationships between documents and concepts. A variation of that is provided in the Lyberworld browsers that show how documents relate to query terms and allow the user to manipulate the relative importance of the concepts to understand better what the retrieved set looks like (15).

One of the more important research and development areas within IR systems is identifying and understanding methods for visualizing data that can be rapidly understood by users. The current state of the art provides several tools for showing results, but work is only beginning toward understanding how those tools can best be applied. Similar work is done in a broader context as part of the field of human-computer interaction.

MULTIMEDIA INFORMATION RETRIEVAL

IR has traditionally been concerned only with text documents, but its nature allows it to extend to other media that represent "text" differently. For example, IR methods can be applied to scanned images of text pages as well as audio recordings of speech. Both approaches rely on methods that convert the image or audio into text, but do so in a way that is robust in the face of the nearly unavoidable errors that occur during the conversion process. Variations of the techniques of IR can also be applied to completely non-textual data such as pictures.

Scanned Documents

The process of converting a scanned document's image into text is called Optical Character Recognition, or OCR. If a document were perfectly converted into ASCII text, then all of IR's techniques would work to retrieve the page images as well as it does for documents. Unfortunately, OCR techniques are rarely error free. An excellent OCR system on very high-quality scanned images of clean pages can achieve an accuracy of over 99%, but 99% accuracy means that on average, there is one character per line incorrectly recognized. On lower quality documents, an accuracy rate of 80–90% or lower is quite common. As discussed above, IR systems are generally based on some form of word matching, so if the OCR process has corrupted enough words, the IR system will not work well.

Fortunately, it turns out that the statistics gathered by IR systems are reasonably robust in the presence of OCR errors (16). As long as the process is not too full of errors, enough of the content-bearing words are correctly recognized to allow the system to work well—errors in words such as the or heretofore are unfortunate but cause few if any problems when searching.

A common variation of IR systems uses character n-grams rather than words as the basic indexing unit. An n-gram is a sequence of characters that are adjacent: "my · cat" contains the 3-grams "my ·," "y · c," " · ca," and "cat." At retrieval, the query is converted into n-grams for comparison. Because an OCR makes errors at the character level, most of the n-grams will be correct in OCR-generated text, so retrieval will be accurate.

An IR system built for OCR systems can also take advantage of extensive statistics that have been gathered about the types of errors that the OCR process makes. These statistics allow a system to recognize the possibility that the non-word "rnen" is probably "men." If used carefully, such adjustments can work around some of the errors that cause retrieval failures.

Recorded Speech

The speech-to-text conversion process is also error prone, though the amount of error depends greatly on the quality of the speech and the recording. Unusual accents, unknown speakers, background noise, and telephone-quality speech all cause failures in the process. Accuracy rates of over 90% are possible with ideal conditions; rates close to 50% are considered good for uncontrolled conditions. An interesting side effect of most speech recognition systems is that when they fail, they produce perfectly valid words (unlike OCR that can produce nonsense words), though they may not be what was said. The recognition systems are also unable to recognize any word that has not been encountered before, meaning that out-of-vocabulary words cause misleading conversions.

The larger error rate and the problem with out-of-vocabulary words mean that different techniques need to be applied for speech. An IR system using the raw speech-to-text output can actually be quite effective, but in situations where new words are regularly being introduced, the IR system must be augmented with more elaborate processing. N-grams are one such technique, though they are less effective here than in OCR.

One interesting approach uses an index that contains the recognized words and a second index that includes all of the individuals work sounds (phones, biphones, and triphones, depending on the setting) in the audio. For example, "find" would result in the phones sequences "f+ay," "f − ay+n," "ay − n+d," "n − d " If a query includes an unknown word, the system can convert it to phone sequences and then use the phone index to look for the word. The two indexes can also be used in combination to increase confidence in the match (17).

Non-textual Pictures

IR is usually used for accessing text; the OCR and speech document examples are merely alternative forms of text. It is possible to apply some of the basic ideas of IR to compare images, also. Text-based IR decomposes documents into low-

level basic features and finds ways to assign them weights according to their significance. The matching process is inexact, recognizing that language understanding is not yet sufficient for more precise techniques to work in general.

In a similar way, image understanding is very inaccurate in a general domain: algorithms for locating and naming objects in an image are not at all robust. However, breaking images into low-level features that are meaningful to users allows images to be compared and retrieved by example. Given a picture of a horse, a system might find other pictures containing horses.

One approach compares two pictures by differences in their color correlation. This information is learned by pairing up colors and counting how often a color appears near another color in the picture. If the color pair correlation of the two pictures resembles each other, then these pictures are said to be close to each other. This technique computes the probability of finding one color pixel at a certain distance from other color pixels. These probabilities are compared for two pictures for all color pairs to compute the distance between two images. This technique is quite robust against size variations and rotations in the two images being compared (18).

Other work derives filters that detect particular patterns of light and dark in text. Some classes of patterns are strongly correlated with image attributes that are meaningful to a person. For example, it is possible to detect texture and so locate grass, sand, trees, and so on. A variation on the method can discover text within an image (e.g., a corporate logo or a billboard) and pull it out so that it can be passed to an OCR system (19).

Image comparison is most useful if it is fast as well as effective. Methods that can build the equivalent of inverted indexes for image retrieval have been developed for some image comparison purposes. This area of work is generating interesting results, but has only begun.

CONCLUSION

At its core, information retrieval studies how to match a user's query against a collection of documents. The field began in the late 1950s but has seen the greatest growth in the last decade as computers became widespread among general users and the cost of disk space for document storage plunged.

Research within the field is based around several models of document meaning, all of which base themselves on using low-level features such as words to capture higher-level meaning. Unlike more elaborate natural language understanding efforts, IR makes no effort to achieve a deep understanding of the text. Interestingly, IR is nonetheless resoundingly successful at retrieving documents in unrestricted settings, a claim that cannot be made by any other text-understanding disciplines.

IR research has traditionally been focused in only a few areas, the most noticeable being ad-hoc and filtering retrieval methods. Development of summarization techniques, effective user interfaces, and nontextual visualizations has been ongoing for many years but has become more important with large document repositories such as the Web. There is much to be learned about how to retrieve documents effectively, but what is already known is already being extended in new directions such as cross-language retrieval and picture comparisons.

Full understanding of text by computer has been often promised but never delivered, and there is no reason to believe that it will be achieved in the near future. Until that distant time arrives, the field of information retrieval offers a highly effective and efficient means for accomplishing nearly the same thing.

BIBLIOGRAPHY

1. F. C. Cheong, *Internet agents: spiders, wanderers, brokers and bots,* Indianapolis, IN: New Riders Publishing, 1995.
2. D. K. Harman, Overview of the fourth Text REtrieval Conference (TREC-4), *Proc. fourth Text Retrieval Conf. (TREC-4),* NIST special publication 500-236, 1–24, 1996.
3. C. J. van Rijsbergen, *Information retrieval* 2nd ed., London: Butterworths, 1979.
4. G. Salton, A. Wong, and C. S. Yang, A vector space model for information retrieval, *Commun. ACM,* **18** (11): 613–620, 1975.
5. G. Salton and C. Buckley, Term-weighting approaches in automatic text retrieval, *Inf. Process. Manage.,* **24** (5): 513–523, 1988.
6. W. B. Croft and D. J. Harper, Using probabilistic models of document retrieval without relevance information, *J. Documentation,* **35** (4): 285–295, 1979.
7. S. E. Robertson, S. Walker, and M. Hancock-Beaulieu, Large text collection experiments on an operational, interactive system: OKAPI at TREC, *Inf. Process. Manage.,* **31** (3): 345–360, 1995.
8. H. Turtle and W. B. Croft, Evaluation of an inference network-based retrieval model, *ACM Trans. Inf. Syst.,* **9** (3): 187–222, 1991.
9. N. J. Belkin and W. B. Croft, Information filtering and information retrieval: two sides of the same coin?, *Commun. ACM,* **35** (12): 29–38, 1992.
10. G. Salton et al., Automatic analysis, theme generation and summarization of machine-readable texts, *Science,* **264**: 1421–1426, 1994.
11. C. D. Paice, Constructing literature abstracts by computer, *Inf. Process. Manage.,* **26**: 171–186, 1990.
12. H. P. Luhn, The automatic creation of literature abstracts, *IBM J. Res. Develop.,* **2**: 159–165, 1958.
13. B. Shneiderman, D. Byrd, and W. B. Croft, Clarifying search: A user-interface framework for text searches, *D-Lib Magazine,* January, 1997.
14. M. A. Hearst and J. O. Pederson, Reexamining the cluster hypothesis: Scatter/Gather on retrieval results, *Proc. 19th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval,* 76–84, 1996.
15. M. Hemmje, C. Kunkel, and A. Willett, LyberWorld—A visualization user interface supporting fulltext retrieval, *Proc. 17th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval,* 249–259, 1994.
16. K. Taghva et al., The effect of noisy data on text retrieval, *J. Amer. Soc. Inf. Sci.,* **45** (1): 50–58, 1994.
17. G. J. F. Jones et al., Retrieving spoken documents by combining multiple index sources, *Proc. 19th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval,* 30–38, 1996.
18. J. Huang et al., Image indexing using color correlograms, *Proc. IEEE Comp. Soc. Conf. Comp. Vision Pattern Recognition (CVPR),* June, 1997.
19. R. Manmatha, Multimedia indexing and retrieval research at the Center for Intelligent Information Retrieval, *Proc. Symp. Document Image Understanding Technol.,* June, 1997.

Reading List

G. Kowalski, *Information Retrieval Systems: Theory and Implementation,* Boston: Kluwer Academic Publishers, 1997.

G. Salton and M. J. McGill, *Introduction to modern information retrieval,* New York: McGraw-Hill, 1983.

G. Salton, *Automatic text processing—the transformation, analysis and retrieval of information by computer,* Reading, MA: Addison-Wesley, 1989.

G. Salton, Improving retrieval performance by relevance feedback, *J. Amer. Soc. Inf. Sci.,* **41** (4), 288–297, 1990.

G. Salton, Developments in automatic text retrieval, *Science,* **253**, 974–980, 1991.

K. Sparck Jones and P. Willett (eds.), *Readings in Information Retrieval,* San Francisco, CA: Morgan Kaufmann, 1997.

I. H. Witten, A. Moffat, and T. C. Bell, *Managing gigabytes: compressing and indexing documents and images,* New York: Van Nostrand Reinhold, 1994

JAMES ALLAN
University of Massachusetts
AMIT SINGHAL
AT&T Labs Research