2005

Document Retrieval, Automatic

Elizabeth D. Liddy
Syracuse University

Recommended Citation

Elizabeth D. Liddy 2005. Automatic Document Retrieval. In Encyclopedia of Language and Linguistics. 2nd Edition. Elsevier Press

Document Retrieval, Automatic

Description/Abstract

Document Retrieval is the computerized process of producing a relevance ranked list of documents in response to an inquirer’s request by comparing their request to an automatically produced index of the documents in the system. Everyone uses such systems today in the form of web-based search engines. While evolving from a fairly small discipline in the 1940s, to a large, profitable industry today, the field has maintained a healthy research focus, supported by test collections and large-scale annual comparative tests of systems. A document retrieval system is comprised of three core modules: document processor, query analyzer, and matching function. There are several theoretical models on which document retrieval systems are based: Boolean, Vector Space, Probabilistic, and Language Model.

Keywords

Document Retrieval, document processing, query processing, Boolean, Vector Space, Probabilistic, Language Model

Disciplines

Library and Information Science | Linguistics

Document Retrieval, Automatic[1]

Elizabeth D. Liddy
Center for Natural Language Processing
School of Information Studies
Syracuse University
www.cnlp.org

Abstract: Document Retrieval is the computerized process of producing a relevance ranked list of documents in response to an inquirer's request by comparing their request to an automatically produced index of the documents in the system. Everyone uses such systems today in the form of web-based search engines. While evolving from a fairly small discipline in the 1940s, to a large, profitable industry today, the field has maintained a healthy research focus, supported by test collections and large-scale annual comparative tests of systems. A document retrieval system is comprised of three core modules: document processor, query analyzer, and matching function. There are several theoretical models on which document retrieval systems are based: Boolean, Vector Space, Probabilistic, and Language Model.

Definition

Document Retrieval (more commonly referred to as Information Retrieval by researchers in the field) is the computerized process of producing a list of documents that are relevant to an inquirer's request by comparing the user's request to an automatically produced index of the textual content of documents in the system. These documents can then be accessed for use within the same system. Nearly everyone today uses Document Retrieval systems, although they may not refer to them as such, but rather as Web-based search engines, e.g. Google, Yahoo, Alta Vista, etc.

Document Retrieval systems are based on different theoretical models, which determine how matching and ranking are conducted. The most prevalent models are Boolean, Vector Space, Probabilistic, and Language Modeling, each of which is explained below. Within the indexing aspect of each model, the system processes, represents, and weights the substantive content of documents and queries for matching. It is here in feature selection that one might expect to see linguistic theories and models used extensively, however, to date, most systems utilize only the morphological and lexical levels of language, with some notable exceptions where full Natural Language Processing is utilized (Liddy, 1998; Strzalkowski et al, 2002).

Following a brief history of the major mileposts and trends in the field of Document Retrieval, the basic components and processes of a generic Document Retrieval System are described in fairly non-technical language in order to provide context for understanding later, more technical distinctions.

History of the Field

Document Retrieval first emerged as a field of both inquiry and application in the late 1940's and early 1950's at a time when the amount of scientific literature being published

[1] Published in the Encyclopedia of Language & Linguistics, 2nd Edition. Elsevier Limited. 2005.

was growing at such a tremendous rate that it raised concerns as to how scientists would be able to stay informed of new developments as they were being reported in the scientific literature. Researchers, mainly computer scientists and library scientists, began to consider how computers might be programmed to accomplish some of the laborious human tasks of representing and retrieving relevant sources that were previously considered within the purview of Librarians and Library Science. Initially, Document Retrieval was actually citation retrieval, that is, a search against bibliographic fields such as title, author, source, and subject-based keywords from a controlled vocabulary that were humanly assigned to documents. The documents themselves were neither actually stored nor accessible by computer. The appropriate analogy is closer to an automated library card catalog search than a full text search of the contents of actual documents. The results from a Document Retrieval search were simply citations that led the user to a hard copy document, book, or report. Over time, advances in the theoretical models underpinning Document Retrieval and decreases in the cost of computer storage permitted the full text of documents to be ingested and searched, not just a limited set of manually assigned bibliographic access points.

While the increase in scientific literature may have been the practical motivator for the field of Document Retrieval, Vannevar Bush's article in the Atlantic Monthly entitled "As We May Think" is considered by most historians of science as the intellectual forbearer of the field (Bush, 1945). In this piece, Bush presaged much of what we today consider to be the crux of the field, when he called for a solution for improved access to information that he referred to as "memex…a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility…an enlarged supplement to his memory."

As all other computer-automated applications at that time, Document Retrieval initially made use of punch cards which enabled the post-coordinate revolution – that is, rather than documents being assigned pre-coordinated descriptors, such as *automatic information retrieval* as would be typical in a card catalog or a printed index, post-coordinate indexing enabled single word descriptors to be assigned one to a card, such as *automatic* on one card describing the document, *information* on another, and *retrieval* on yet another card and then for the user to combine search terms as they best saw fit to describe their information need at search time.

The increasing availability of machine readable texts led to rapid, widespread growth in the usage of Document Retrieval systems as the collection against which users could search increased dramatically. Not surprisingly, this also led to the flourishing of the early commercial Document Retrieval systems such as Dialog and LexisNexis. These systems were sufficiently complex and non-intuitive that end users needed to have trained search intermediaries do their searches, because these intermediaries could understand the intricacies of the data records in the Document Retrieval System and were well-trained in constructing queries in Boolean logic. These early systems were not based on free text searching, but rather, required the intermediary to know the exact wording and syntax to use in searching, both in proper names and subject based descriptions – referred to respectively as authority files and controlled vocabulary.

While the commercial realizations of Document Retrieval were seeing success in improved provision of citations or actual documents to researchers in the scientific disciplines, researchers in the growing field of Information Science were actively moving on to more complex models of retrieval and more natural representations of documents on which users themselves might search – namely the text as it occurred naturally in the documents. Of course this then obviated the need for intermediaries, lowered the cost of entry to the field, and opened the modes of access. This was a trend that began in the early days of Document Retrieval, but which really only came to full fruition with the tremendous growth in open availability of information on the Web. Interestingly, the Document Retrieval systems, with full-text searching, relative weighting of terms, and ranking of results, which were being developed and tested with increasing rigor by academic researchers, were **not** adopted by commercial providers of Document Retrieval services, due to these organizations' sizeable investments in Boolean-based retrieval.

To the growing number of academic computer science departments and research labs that were becoming interested in Document Retrieval, the Cranfield experiments that ran from 1957 to 1967, (Cleverdon, 1967) were a major milestone as they established the basic evaluation paradigm, including metrics and experimental procedures which enabled true scientific experimentation to be conducted and the field to be recognized as a science. The test collection paradigm of gathering a stable collection of documents of interest to an identifiable user group, as well as possible questions from these users that might be expected to be answerable by documents in that collection, and relevance assessments on the retrieved documents (or the whole collection if possible) is still used today. The accepted metrics used in Document Retrieval experimentation and evaluation were also established, namely, Recall – the percent of known relevant documents in the collection that are retrieved; and Precision - the percent of retrieved documents that are relevant.

Based on the Cranfield test collections and the new evaluation metrics, an era of great research activity and development ensued. However, unlike Cranfield's work which evaluated automatic searching of manually assigned index terms, the ensuing work utilized these same collections for experimentation on automatic indexing of the natural language of document abstracts first, and later, of the full text of the documents. The various models of Document Retrieval which are so well-known and experimented with today were introduced, including the vector space model of Salton and his group of researchers at Cornell (Salton, 1968); the probabilistic model (Robertson & Sparck Jones, 1976); and the inference model (Turtle & Croft, 1990). The most widely known and used of the experimental systems was SMART, developed by Salton (1971). It was used for extensive empirical runs which showed the value of the simple natural language processing techniques of stemming, deletion of stop words, phrase-based indexing, as well as many experiments investigating the most appropriate term-weighting formula.

Document Retrieval systems with empirically-tested ranking and weighting algorithms slowly spread into the operational world as increasing numbers of experimental results demonstrated that automatic indexing of the natural language of documents was as good as manually controlled vocabulary indexing. The range of retrieval models that were utilized commercially increased due to availability of increased computing power and lower cost of storage of indexes based on the full text of documents.

Large scale comparative testing of Document Retrieval systems began with the first of the Text REtrieval Conferences (TREC) at the National Institute of Standards and Technology in 1992 (Harman, 1993) and has continued and expanded each year. The goal of these annual events is to bring together researchers from around the world who test their systems on a common test collection of queries and documents and then share the details of their systems with the other participants. Twenty-five research groups participated in the first evaluation using a large, new test collection, a set of questions generated by a likely group of users, and their relevance assessments on the documents retrieved by the participating systems. Most member of the Document Retrieval community believe that TREC has been one of the most positive influences on both scientific advances and expansion of interest in Document Retrieval.

However, most would also agree that it is the Web that has made the field as prominent and exciting as it is today. Web searching is now omnipresent in most individuals' lives and little thought is given to it as being the original Document Retrieval that began back in the 1940s. And while there are some real differences between earlier methods and practices of Document Retrieval from pre-established databases as compared to the very dynamic and less controlled world of Web searching, the basics are the same, with the addition of link analysis, which utilizes a network structure of links among web pages as an additional source of information (in addition to document content) in retrieving and ranking potentially relevant documents or pages.

Walk-Through of a Basic Document Retrieval System

In Document Retrieval, some processes take place dynamically when the user inputs their query, while other processes take place off-line in advance and in batch mode and do not involve individual users. These static processes are run on the documents that will be made available in the retrieval system. These will be explained first. Then, the two dynamic processes, Query Processing and Matching, will be presented. Figure 1 provides a simple, but clear view of the relationship between these three processes.

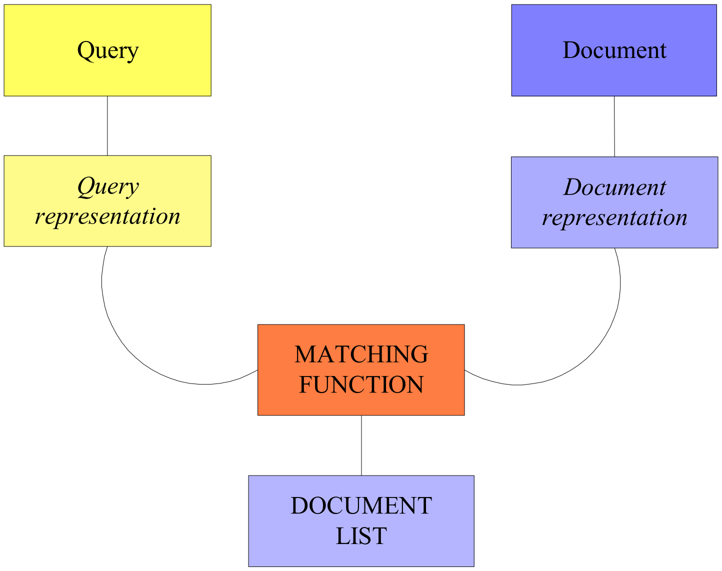

Figure 1: Basic Components of a Document Retrieval System

Document Processing: The first two steps in the processing of documents are somewhat mundane, but necessary, and can be considered as batch pre-processing. These are:

1. **Normalize document stream** to a predefined format, whereby multiple external formats (e.g. newsfeeds, web pages, word processed documents) are standardized into a single consistent format. This is an essential step (much akin to data clean-up in data mining) as all downstream processes rely on receiving a common format they can recognize and process. Preprocessing is particularly vital for systems with more complex processing than simple '*characters between white spaces*' indexing.

2. **Break document stream into desired retrievable units**, whether this is web page, chapter, full document, paragraph, etc. The pointers stored in the inverted file are to whatever unit size has been pre-determined. Therefore, document retrieval could in fact be paragraph retrieval, if the indexable unit was determined at this stage to be the paragraph.

From this step forward, the system is performing the heart of the document indexing process.

3. **Identify potential indexable elements in documents.** This is a key decision point that dramatically affects the nature and quality of the retrieval performance. First, the important definition needs to be made as to what is a term. Is it any string of alphanumeric characters between blank spaces or punctuation? If so, are non-compositional phrases or multi-word proper names, or inter-word symbols such as hyphens or apostrophes treated differently (e.g. are "small business men" and "small-business men" the same)? At this stage, the system requires a set of rules to be executed which control what actions are taken by the 'tokenizer' – the algorithm which recognizes 'indexable terms'. IR systems vary as to which of these processes they perform, but the most frequently used processes are:

    a. **Delete stop words** via an algorithm that filters the document's potential indexable elements against a Stop Word list to eliminate terms that are deemed to be insignificant in determining a document's relevance to a user's request. The original objective in using stop words was to save system resources by eliminating those terms that have little value for retrieval performance. Although these terms may comprise up to 40% of the tokens in a document set, index size is of far less importance today due to cheap memory, but their omnipresence renders them of little value to retrieval. The typical word classes that are marked as stop words include the function word classes and a few more (i.e. articles, conjunctions, interjections, prepositions, pronouns, and 'to be' verb forms).

b. **Stem terms by removing suffixes**. In this morphological step, some IR systems do just inflectional ('weak') stemming which only changes the subclass within a part-of-speech category, i.e. past tense to present tense, while others also do derivational ('strong') stemming which removes suffixes, sometimes recursively, that may actually change the part of speech of a word.

   Use of stemming will result in fewer entries in an index, each of which is likely to have higher frequency counts than if all morphological variants and their counts are used. The initial goal of stemming was to reduce the storage requirements of the inverted index file by reducing the number of unique words, but stemming has remained in use even today when storage is not an issue, because it improves recall of relevant documents. For example, if a query includes *analyze*, the user may well want documents which contain *analysis*, *analyzing*, *analyzer*, or *analyzed*. In order for the system to match on all these variants, it must stem both the query and the document terms to *analy-*. Obviously, stemming may negatively impact precision.

c. **Bracket noun phrases**, usually by means of regular expressions which define the part-of-speech patterns which comprise a noun phrase (e.g. <ADJ NN> or <NN NN>). This is a step that can negatively affect recall of retrieval results by either excluding documents when the phrasal expression in the query is not exactly the same as the index entry of a document, or positively affect precision by retrieving only documents that include the terms in the desired phrasal expression.

4. **Produce an inverted file** containing a sorted array of all indexable terms (with term defined as referring to either a word or a phrase), along with the unique identification number of each document in the collection in which the term occurs, a link to each of these documents, weights for each term as determined by the IR model being implemented in the system [which will be described in the next section] and optionally, the within-document location of the term. More sophisticated systems may include further information in the inverted file, such as named entity category for Proper Names (i.e. PERSON, ORGANIZATION, GEO-LOCATION, etc) but the most common features are simply term, document ID, and weight.

Query Processing: The system's internal representation of the user's question / search terms is typically referred to as the query. Most of the same processes that are run on the documents are also run to produce the query, but there are some unique processes as well. As distinct from document processing, all of the query processing is done in real time, while the user awaits their documents. These are:

1. **Recognize query terms vs. special operators,** such as "*I need information about…*" which do not convey the topic of the user's information need and will not be included in the query representation**.**

2. **Tokenize query terms,** a process that requires similar decisions as were described on the document processing side – that is stop word deletion, stemming, and phrase recognition.

3. **Create query representation,** which typically follows stop word removal and stemming, and which may also include insertion of logical operators between / amongst terms requiring co-occurrence or simple presence of only one of the arguments.

4. **Expand query terms** to include variant terms that refer to or relate to the same concept. These may be synonymous terms that are found in an electronic thesaurus such as WordNet (Fellbaum, 1998) or terms that are highly associated with the query term, based on co-occurrence statistics preferably computed on the same or a similar document collection as the one on which the search is being conducted. Query expansion relieves the user of needing to generate all conceptual variants of their search terms and is likely to improve recall, but may reduce precision when erroneous senses of the newly introduced terms retrieve irrelevant documents. The longer a query is, the less likelihood that erroneous senses of expanded terms will have a negative impact, but also the less likely that expansion will contribute much to the retrieval results**.**

5. **Compute query term weights.** This step is less commonly included in Document Retrieval systems, mainly because it is difficult both for users to know how to assign weights to query terms in a way that improves retrieval results, or for automatic weighting, since queries are frequently so short as to give little evidence of the relative importance of query terms as most terms only occur once in a single query. Some NLP-based systems have positive results from automatic determination of the 'mandatory' concept in a query which is then assigned a greater weight (Liddy et al, 1995).

Matching of Query to Documents: Once the query representation is produced, the matching process begins. The process description below may be easier to follow if you conceive of both the query and the documents as vectors of terms, with frequency information or weights for each term in the vector.

1. **Search inverted file** for documents that contain terms in the query. This is typically done using a standard binary search. Each document that contains any of the query terms becomes a candidate for retrieval.

2. **Compute similarity score** between query and each candidate document using the algorithm prescribed by one of the four Document Retrieval models being used. This score is referred to as the Similarity Coefficient. The scoring mechanism for each of the major Document Retrieval models will be detailed in the next section.

3. **Rank order the documents** in decreasing order based on the scores assigned them by the scoring algorithm. This may be either straightforward ranking based on the

Similarity Coefficient, or the system may utilize automatic relevance feedback whereby the system takes the top N-ranked terms from the top N-ranked documents as they are being shown to the user, and adds these terms to the query representation and reruns the search with the revised query to produce the continuation of the ranked list of relevant documents.

4. **Provide list of perceived relevant documents to user** ranked by similarity score between query and document. Systems that utilize other sources of evidence of value of a document to the query, such as number of links from the page/document to or from other pages/documents, would integrate this information and produce a potentially different ranked list.

5. **Allow for query modification by the user** if user-based relevance feedback is provided by the system. If so, typically, the user marks the documents they find relevant, either based on just the title and brief description shown them in the initial list or by actually reviewing the full document, which they can link to from the results page.

6. **Perform relevance feedback based on user's input**. The algorithm for user-based relevance feedback is typically the same as that for automatic relevance feedback as described in Step 3 above. The system then re-runs the search with the revised, and hopefully improved, query and produces a revised ranked list of documents. The relevance feedback loop is iterative and can be performed as many times as the user wants.

Theoretical Models

Over the years, a number of different theory-based models have provided the basis of implemented Document Retrieval systems. Four of the most influential models are briefly presented here – Boolean, Vector Space, Probabilistic, and Language Models – in the order in which they have come into use in the field. Mathematical details of the models as well as formulas for their implementation are beyond the scope of this piece, and can be found in Information Retrieval textbooks such as Baeza-Yates & Ribeiro-Neto (1999).

The Boolean Model is the earliest and the easiest to understand of the four models. And, counter to the above description of the typical representation of documents in an inverted file, Boolean systems do not use either term frequencies or term weights. Terms are simply present or not in a document. As a result, the inverted file is smaller and easier to manage and update. The Boolean operators – AND, OR, and NOT – are used both for representing the user's need and in the internal matching of the query to the inverted file. For example, if a user is interested in "*the relative efficiency of oil vs. gas*", their Boolean query would be (*efficiency* AND (*oil* OR *gas*)), assuming that *relative* is dropped from the query because it is on the Stop Word List. Only documents that have either *oil* and *efficiency* or *gas* and *efficiency* would be retrieved. But, since the Boolean Model does not use weights, the order in which the documents are presented to the user cannot be based on system-predicted degree of relevance. Therefore they are presented based on another dimension – typically date accessioned into the system.

While the Boolean model may appear crude as compared to more sophisticated models, it has the advantages of ease and efficiency of the implementation, simplicity of the inverted file structure, and clarity of the model. Disadvantages of the Boolean model are its inability to relevance rank documents and the fact that users find Boolean queries difficult to produce as the logical interpretations of the Boolean operators do not reflect their typical use in natural language. Attempts have been made to deal with both of these issues. The first by the introduction of the extended Boolean Model by Salton, Fox, and Wu (1983) which relaxes the logical interpretation of the Boolean operators by using them as distances and thereby providing relevance ranked lists of retrieved documents. The second acknowledge disadvantage of the Boolean model has been addressed by the development of user interfaces that either clarify the Boolean logic, or mask it from the user. The latter is done by allowing users to ask their questions in straightforward natural language from which a Natural Language Processing module in the system produces the Boolean-structured query automatically based on the natural logic embedded in a language's syntax and semantics.

The **Vector Space Model** (Salton, 1971) is the most commonly used model in Document Retrieval Systems today due to its consistent, proven performance across multiple implementations on many collections. Conceptually, in the vector space model, a document is represented by a vector of the terms in the document, and these vectors exist in term space, which is the size of all the unique terms in the collection. Each term represents a dimension in this term space and the similarity between a query and a document is measured by the closeness of the query vector and the document vector, where closeness is measured by the angle between the two vectors. Either cosine similarity or the inner / dot product is used to compute the angle between vectors.

However, to compute a similarity score between query and documents in the collection and then rank order the documents based on their likely relevance to the query, it is typical to use weighted vectors. The system needs a basis on which to assign weights to both query and document terms. While there are multiple ways to compute such weights, the nearly universal way to do this is what is known as tf / idf – that is term frequency (tf) multiplied by inverse document frequency (idf) (or equivalently, divided by document frequency). By use of this weighting scheme, the Vector Space Model is saying that the best indexing terms are those that occur with high frequency in a document (tf) relative to their occurrence in other documents in the collection (idf). The tf metric is considered an indication of how well a term characterizes the content of a document. Of course, there are several arguments that might be made against this view, such as the linguistic phenomena of synonymy and anaphora – both of which can represent the same concept with different terms, thereby resulting in the candidate index term undercounting conceptual presence. The idf, in turn, reflects the number of documents in the collection in which the term occurs, irrespective of the number of times it occurs in those documents. Using these two metrics in combination, a query-to-document similarity score is computed between a query and each document in the collection. Based on these similarity scores, the model produces a ranked list of documents in terms of predicted relevance to the query.

Computing tf and idf requires first determining which features (e.g. words, phrases) will be used in representing documents and queries. It might be of interest to linguists that until recently, relatively little attention was paid to what these features were. While noun phrases were historically used as subject headings in library catalogs and controlled vocabularies, doing this automatically requires the ability to distinguish noun phrase elements from other parts of speech, which was beyond the state of the art when Document Retrieval Systems were first introduced. But, the belief that some words provide a better representation of documents and queries led to the use of a Stop Word List which excludes closed class terms (e.g. prepositions, pronouns, determiners) from indexing, thereby leaving nouns, adjectives, verbs, and adverbs as the feature set.
Advantages of the Vector Space Model are that, distinct from the Boolean Model, it allows partial matching of query and document, and the model's easy adaptability via adjustments to its parameters, including term-weighting schemes, which have been shown to have a major impact on the quality of retrieval results. As mentioned above, the Vector Space Model's performance is consistently good with general collections. The disadvantages of the Vector Space Model include its weighting schemes' reliance on information from across the database, and the need therefore for weights to be updated as the database changes. However, research has shown that less frequent updating of collection figures does not negatively impact performance significantly if the collection is large enough.

The **Probabilistic Model**, which was first introduced by Robertson and Sparck Jones (1976), is based on several assumptions, most significantly, that terms occur independently of each other in the documents. Basically, using Bayes Theorem, a Document Retrieval System built on the Probabilistic Model assigns the odds of relevance for each term in a document based on that term's frequency in a set of known relevant documents. This appears to require the use of a set of known relevant and known non-relevant documents in order to estimate the relevance probability for a new document. But, given that it is unlikely, without relevance feedback information being available, that relevance is known a priori, the odds that a term will appear in a relevant document is initially estimated by counting the documents in which the term appears and the documents in which it does not appear. This initial estimate is then adjusted based on information from the user, which results in a refinement of the estimate. This process of refinement can occur without input from the user, based on the distribution of the query term in the documents retrieved, which will be considered as relevant, while the documents that are not retrieved are considered non-relevant.

Over the years, a number of weighting formulas have been developed and tested for use within the Probabilistic Model, starting with the Robertson-Sparck Jones BM1 version (1976) up to the weighting and document ranking function known as Okapi BM25 (Robertson et al, 1998), which is used in many experimental text retrieval systems, with more tuning parameters.

The advantages of the Probabilistic Model are its solid theoretical foundation and the fact that the model performs well in general collection experiments. Also valued is its ability to rank documents in terms of their probability of relevance. Its disadvantages are its problematic assumption that probabilities are based on a binary condition of relevance, the assumption of term independence, which is not realistic, and the lack of actual relevance data in the system's initial assignments.

The last model, the **Language Model** was first introduced into Document Retrieval by Ponte & Croft in 1998 and has become an increasingly popular model in Document Retrieval research, based as it is on many years of research in statistical language modeling for successful applications such as speech recognition and spelling correction. Language Modeling is a statistical method for ranking documents in a collection based on the probability that they might have generated the query (see Language processing, statistical methods). An intuitive explanation of language modeling for IR is that each document has its own language model and queries are thought of as being generated by a document language model. So the operative question is: *"Which document(s) would produce this query?"* The documents in the collection are evaluated and ranked based on the probability of their language model generating that query. It is interesting to note that the system here is described as predicting probability of query production, not probability of relevance of the document, and has become known as the query-likelihood retrieval model.

The basic language model shares many of the same theoretical assumptions of other probabilistic models, and utilizes n-gram analysis and a hidden Markov model. To establish the word n-gram language model, probability estimates are derived from n-gram frequencies in the training data and then smoothing techniques are used to estimate frequencies for those words that do not actually occur in the training data. The Language Model has been extended in various ways by a number of research groups as discussed in detail in Liu & Croft (2004). It is also believed that the use of a more linguistically sophisticated document representation may even further improve the performance of the basic language model (Hiemstra, 2000).

References

Baeza-Yates, R. & Ribeiro-Neto, B. (1999). Modern Information Retrieval. Reading, MA., Addison-Wesley.

Bush, V. (1945). As We May Think. *Atlantic Monthly*. Vol. 176(1), 101-108.

Cleverdon, C.W. (1967). The Cranfield Tests on Indexing Language Devices. *Aslib Proceedings*, 19, 173-92.

Fellbaum, C. (Ed). (1998). WordNet: An Electronic Lexical Database. Cambridge, MA. MIT Press.

Harman, D. (1993). Overview of the First TREC Conference. Proceedings of ACM-SIGIR-93 Conference. Pittsburgh, PA. 36-47.

Hiemstra, D. (2000). Using Language Models for Information Retrieval. Enschede, The Netherlands, Neslia Paniculata.

Liddy, E.D. (1998). Enhanced Text Retrieval Using Natural Language Processing. Bulletin of the American Society for Information Science. Vol 24, No. 4. http://www.asis.org/Bulletin/Apr-98/liddy.html

Liddy, E.D., Paik, W., McKenna, M. & Yu, E.S. (1995). A natural language text retrieval system with relevance feedback. Proceedings of the 16th National Online Meeting.

Liu, X. & Croft, W.B. (2004). Statistical Language Modeling for Information Retrieval. In Cronin, B. (Ed.). Annual Review of Information Science & Technology. Vol 38.

Ponte, J. & Croft, W.B. (1998). A Language Modeling Approach to Information Retrieval. In Proceedings of the 21st ACM Conference on Research and Development in Information Retrieval.

Robertson, S.E. & Sparck Jones, K. (1976). Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences*. 27(3): 129-46.

Robertson, S.E., Walker, S. & Beaulieu, M. (1998). Okapi at TREC-7. In Seventh Text REtrieval Conference (TREC-7), Gaithersburg, MD.

Salton, G. (1968). Automatic Information Organisation and Retrieval. New York: McGraw-Hill.

Salton, G. (1971). The SMART Retrieval System. Englewood Cliffs, NJ: Prentice Hall.

Salton, G., Fox, E. & Wu, H. (1983). Extended Boolean Information Retrieval. *Communications of the ACM*. 26(11). 1022-36.

Sparck Jones, K. & Willett, P. (Eds). (1997). Readings in Information Retrieval. San Francisco, Morgan Kaufmann Publishers.

Strzalkowski, T., Tzukermann, E. & Klavans, J. (2002). Information Retrieval and Natural Language Processing. In Mitkov, R. (Ed.), Handbook of Computational Linguistics, Oxford University Press.

Turtle, H. & Croft, W.B. (1990). Inference Networks for Information Retrieval. In Proceedings of the 13th Annual International ACM SIGIR Conference, Brussels, Belgium, pp. 1-24.