4

Searching for Text Documents

Henk Blanken and Djoerd Hiemstra

University of Twente

4.1 Introduction

Many documents contain, besides text, also images, tables, and so on. This chapter concentrates on the text part only. Traditionally, systems handling text documents are called information storage and retrieval systems. Before the World-Wide Web emerged, such systems were almost exclusively used by professional users, so-called indexers and searchers, e.g., for medical research, in libraries, by governmental organizations and archives. Typically, professional users act as "search intermediaries" for end users. They try to figure out in an interactive dialogue with the system and the end user what it is the end user needs, and how this information should be used in a successful search. Professionals know the collection, they know how documents in the collection are represented in the system, and they know how to use Boolean search operators to control the number of retrieved documents.

Many modern information retrieval systems, like Internet search engines, are specifically designed for end users who are not familiar with the collection, the representation of the documents, and the use of Boolean operators. The main requirements for these systems are the following. Firstly, users should be able to enter any natural language word(s), phrase(s) or sentence(s) to the system without the need to enter operators. Secondly, the system should rank the retrieved documents by their estimated degree or probability of usefulness for the user.

In this introduction we will reconsider some concepts from previous chapters and describe what these concepts mean in the information retrieval realm.

4.1.1 Text Documents

A (text) document has an identification and can be considered to be a list of words. So, a book is a document, but so is a paper in the proceedings of a conference or a Web page. The identification may be an ISBN number for a book, the title of the paper together with the ISBN of the conference proceedings or a URL for a Web page.

Retrieval of text documents does normally not imply the presentation of the whole document (this is too space and time consuming). Instead the system presents the identifications of the selected documents possibly together with brief descriptions and/or their rankings.

4.1.2 Indexing

Indexing is the process of deriving metadata from documents and storage of the metadata in an index. The index describes in one way or another the content of the documents; for text documents the content is described by *terms* like `social` or `political`. During retrieval, the system uses the index to determine the output.

There are two ways to fill the index, namely manually and automatically. Professional users like librarians may add so-called *assigned terms* to documents as a kind of annotation. Sometimes these terms are selected from a prescribed set of terms, the *catalog*. A catalog is composed by specialists and describes a certain (scientific) field. An advantage of this approach is that the professional users know the allowed terms to be used in query formulation. A clear disadvantage is the amount of work needed to perform the manual indexing process.

Describing the content of documents can also be done automatically resulting in so-called *derived terms*. Several steps are required, for instance a step in which an algorithm identifies words in an English text and puts them to lower case. Other steps use basic tools like *stop word removal* and *stemming*. Stop words are words in the document with little meaning, mostly function words like “the” and “it”. These words are removed. Stemming conflates the words in the document to their stem. For instance, the stemmer introduced by Porter [23] conflates the words “computer”, “compute” and “computation” to the stem `comput`.

4.1.3 Query Formulation

The process of representing the information need is often referred to as the *query formulation* process. The resulting formal representation is the query. In a broad sense, query formulation might denote the complete interactive dialogue between system and user, leading not only to a suitable query but possibly also to a better understanding by the user of his/her information need. Here, however, query formulation denotes the formulation of the query when there are no previously retrieved documents to guide the search, that is, the formulation of the initial query.

Again we must distinguish between the professional searcher and the casual end user. The first one knows the document collection and the assigned terms. The professional will use Boolean operators to compose the query and will be able to adequately rephrase the query depending on the output of the system. If the result set is too small, the professional must broaden the query, if too large the professional must make the query more restrictive. See Section 4.2.

The end user likes to communicate the need for information to the system in natural language. Such a natural language statement of the information need is called a request. Automatic query formulation includes receiving the request and generating an initial query by applying the same algorithms as used for the derivation of terms. The query consists in general of a list of query terms. The system accepts this list and composes in one way or another a result set. The end user may indicate the documents that are considered to be relevant. This *relevance feedback* allows the system to formulate a successive query.

4.1.4 Matching

Probably the most important part of an information retrieval system is the matching algorithm. The algorithm compares the query against the document representations in the index. We distinguish *exact matching* and *inexact matching* algorithms. To start with the first kind: a Boolean query formulated by a professional searcher defines exactly the set of documents that satisfy the query. For each document the system generates a yes/no decision.

If a system uses inexact matching, it delivers a ranked list of documents. Users will walk down this document list in search of the information they need. Ranked retrieval will hopefully put the relevant documents somewhere in the top of the ranked list, minimizing the time the user has to invest on reading the documents. Simple but effective ranking algorithms use the frequency distribution of terms over documents. For instance, the words "family" and "entertainment" mentioned in a small part of a book, may occur relatively infrequent in the rest of the book, which indicates that the book should not receive a top ranking for the request "family entertainment". Ranking algorithms based on statistical approaches easily halve the time the user has to spend on reading documents. The description of ranking algorithms is a major theme of this chapter.

4.1.5 Relation to Other Chapters

If a document contains text and for instance images, then an algorithm needs to separate those parts. In Chapter 3 approaches are described to deal with this problem.

In Section 4.3.6 we deal with the PageRank algorithm used in the Google Web search engine. The algorithm takes into account the hyperlink structure on the Web and has some similarities with collaborative filtering that is explained in Chapter 11. In this technique the opinion of users regarding documents influences the selection process.

4.1.6 Outline

In the traditional information retrieval systems, which are usually operated by professional searchers, only the matching process is automated; indexing and query formulation are manual processes. These information retrieval systems use the Boolean model of information retrieval. The Boolean model is an *exact matching model*, that is, it either retrieves documents or not without ranking them. The model also supports the use of structured queries, which do not only contain query terms, but also relations between the terms defined by the query operators AND, OR and NOT. In Section 4.2 we explain the Boolean model.

In modern information retrieval systems, which are commonly used by non-professional users, query formulation is also automated. Mathematical models are used to model the matching process. There are many candidate models for the matching process of ranked retrieval systems. These models are so-called *inexact matching models*, that is, they compute a ranking for each document retrieved even if the document only partly satisfies the query. Each of these models has its own advantages and disadvantages. However, there are two classical candidate models for approximate matching: the vector space model and the probabilistic model. In Section 4.3 we explain these models as well as other ranking models like the p-norm extended Boolean model, and the Bayesian network model.

So-called language models were first used in telecommunications and some time later in speech recognition. Language models build a mathematical model of a language. This model can be used for instance to determine the probability that a certain word follows a recognized word. Recently the models got much attention in the IR community. Language models are treated in Section 4.3.5.

Web search engines are a rather new phenomenon and the most successful engine is probably Google. Besides some content oriented ranking techniques, Google also exploits the so-called PageRank algorithm. The idea is that the opinion of the user community with respect to a document, that is Web page, influences the ranking of the page. The opinion is modeled by considering the reference pattern which can be derived from the Web. Section 4.3.6 discusses Google's ranking mechanism.

Until now terms are treated equally in a query and in the document as represented in the index. Much attention in IR research has been paid, however, to so-called term weighting algorithms. A *term weight* is a value of the term's importance in a query or a document. Term weighting is described in Section 4.4.

4.2 Boolean Model

The Boolean model is the first model of information retrieval and probably also the most criticized model. The model is based on set theory and can be explained by thinking of a query term as an unambiguous definition of a set of documents. For instance, the query term `economic` simply defines the set of all documents that are indexed with the term `economic`. Using the operators of George Boole's mathematical logic, query terms and their corresponding sets of documents can be combined to form new sets of documents. Boole defined three basic operators: AND, OR, and NOT [4]. Combining terms with the

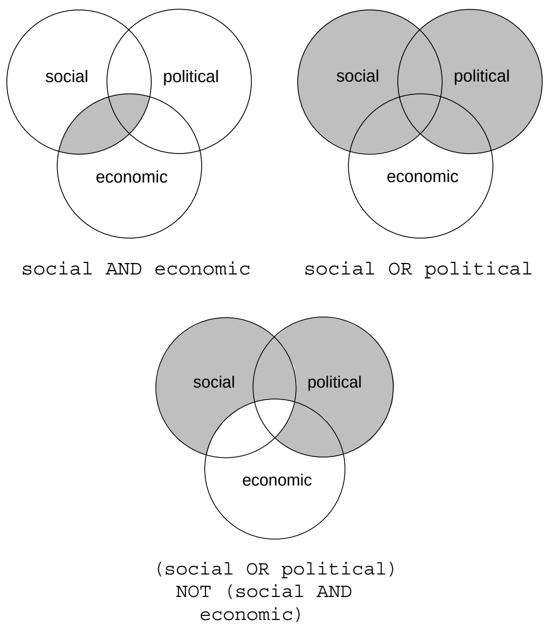


Fig. 4.1. Three Boolean combinations of sets visualized as Venn diagrams.

AND operator will define a document set that is smaller than or equal to the document sets of any of the single terms. For instance, the query `social AND economic` will produce the set of documents that are indexed both with `social` and `economic`. Combining terms with the OR operator will define a document set that is bigger than or equal to the document sets of any of the single terms. So, the query `social OR political` will produce the set of documents that are indexed with either `social` or `political`, or both.

This is visualized in the Venn diagrams of Figure 4.1[1] in which each set of documents is visualized by a disk. The intersections of these disks and their complements divide the document collection into eight non-overlapping regions, the unions of which give 256 different Boolean combinations of social, political and economic documents. In Figure 4.1, the retrieved sets are visualized by the shaded areas.

4.2.1 Proximity Searching: ADJ, NEAR

With the emergence of *automatic full text indexing* (meaning that every word of the document is indexed), commercial retrieval systems added new Boolean operators to the standard Boolean operators. These operators use positions

[1] Often, the NOT-operator is implemented as a logical difference instead of a set complement, requiring the use of `A NOT B` instead of `A AND NOT B`

of words in text. The ADJ operator allows for the search of exact phrases by looking for documents that contain two adjacent terms in the specified order. For instance, `environmental ADJ damage` selects only documents containing the exact phrase "environmental damage". The NEAR operator allows for the search of two terms that are near to each other without any requirements on the order of the words. Table 4.1 list some examples.

Table 4.1. Proximity operators.

Query	Interpretation
`waste ADJ management`	select documents containing the exact phrase "`waste management`"
`waste NEAR management`	select documents containing, e.g., "`waste management`", "`manage-ment of waste`" or "`waste of valuable management talent`"
`(hazardous OR toxic) ADJ wastes`	select documents containing either "`hazardous wastes`" or "`toxic wastes`"
`(hazardous AND waste) ADJ management`	ill-defined because "`management`" could not be adjacent to both "`hazardous`" and "`waste`"

The last example of Table 4.1 requires some explanation. Some systems produce an error if such a query is entered as these systems claim that it is impossible that a term `management` is adjacent to two other terms `hazardous` and `waste`. But `management` may occur many times in a document and usually system designers decide to process the last example as `(hazardous ADJ management) AND (waste ADJ management)`.

4.2.2 Wildcards

Wildcards are used to mask a part of a query term with a special character, allowing it to match any term that maps to the unmasked portion of the query term. Table 4.2 shows some examples of the use of wildcards, taken from Kowalski [14].

From the options in Table 4.2, suffix searches are the most common. In some systems suffix searches are the default without the user having to specify this. Suffix truncation is also the easiest of the options above to implement.

4.2.3 Discussion

We give two advantages of Boolean retrieval. Firstly, the model gives (expert) users a sense of control over the system. It is immediately clear why a

Table 4.2. Wildcards.

Query	Interpretation
`dog*`	suffix truncation selects documents containing, e.g., "`dog`", "`dogs`" or "`doggy`", but also "`dogma`" and "`dogger`"
`*computer`	prefix truncation: selects documents containing, e.g., "`minicomputer`", "`microcomputer`" or "`computer`"
`colo*r`	infix truncation: selects documents containing, e.g., "`colour`", "`color`", but also "`colorimeter`" or "`colourbearer`"
`multi$national`	single position truncation: selects documents containing "`multi-national`" or "`multinational`", but no "`multi national`" as it contains two processing tokens

document has been retrieved given a query. If the resulting document set is either too small or too big, it is directly clear which operators will produce respectively a bigger or smaller set. Secondly, the model can be extended with proximity operators and wildcard operators in a mathematically sound way, which makes it a powerful candidate for full text retrieval systems as well.

We also give two disadvantages of the Boolean model starting with its inability to rank documents. For this reason, the model does not fit the needs of modern full text retrieval systems like for instance Web search engines. On the Web, and for many other full text retrieval systems, ranking is of utmost importance.

A second disadvantage is that the rigid difference between the Boolean AND and OR operators does not exist between the natural language words "and" and "or". For instance, someone interested in "social" *and* "political" documents, should enter the query `social OR political` to retrieve all possibly interesting documents. In fact, the Boolean model is more complex than the real needs of users would justify. Expert users of Boolean retrieval systems tend to use faceted queries. A *faceted query* is a query that uses disjuncts of quasi-synonyms: the facets, conjoined with the AND operator. The following query for instance has two facets: `(economic OR financial OR monetary) AND (internet OR www OR portal)`.

4.3 Models for Ranked Retrieval

The Boolean model's inability to rank documents is addressed by the models presented in this section. A key issue of models of ranked retrieval is automatic query formulation. Non-expert users should be able to enter a request in natural language, or possibly just a couple of terms, without the use of operators. Both ranking and the fact that operators are not mandatory is shared by the approaches presented in this section. Pros and cons are identified for each model.

4.3.1 The Vector Space Model

Luhn [16] was the first to suggest a statistical approach to search for information. He suggested that in order to search in a document collection, the user should first prepare a document that is similar to the needed documents. The *degree of similarity* between the representation of the prepared document and the representations of the documents in the collection are used to search the collection.

Salton [30] found a nice theoretical underpinning of Luhn's similarity criterion. They considered the representations of the documents in the index and the query as vectors embedded in a high-dimensional Euclidean space, where each term is assigned a separate dimension. The document's index representation is a vector $\mathbf{d} = (d_1, d_2, \cdots, d_m)$ of which each component d_k $(1 \leq k \leq m)$ is associated with an index term, while the query is a similar vector $\mathbf{q} = (q_1, q_2, \cdots, q_m)$ of which the components are associated with the same terms.

The similarity measure is usually the cosine of the angle that separates the two vectors $\mathbf{d}$ and $\mathbf{q}$. The cosine of an angle is 0 if the vectors are orthogonal in the multidimensional space and 1 if the angle is 0 degrees:

$$\text{score}(\mathbf{d}, \mathbf{q}) \quad = \quad \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2} \cdot \sqrt{\sum_{k=1}^{m} (q_k)^2}}. \tag{4.1}$$

The metaphor of angles between vectors in a multidimensional space makes it easy to explain the implications of the model to non-experts. Up to three dimensions, one can easily visualize the document and query vectors. Figure 4.2 visualizes an example document vector and an example query vector in the space that is spanned by the three terms `social`, `economic` and `political`.

Relevance Feedback

Measuring the cosine of the angle between vectors is equivalent with normalizing the vectors to unit length and taking the vector inner product:

$$\text{score}(\mathbf{d}, \mathbf{q}) \quad = \quad \sum_{k=1}^{m} n(d_k) \cdot n(q_k) \qquad \text{where } n(v_k) = \frac{v_k}{\sqrt{\sum_{k=1}^{m} (v_k)^2}}. \tag{4.2}$$

Some rather *ad hoc*, but quite successful retrieval algorithms are nicely grounded in the vector space model if the vector lengths are normalized. An example is the relevance feedback algorithm by Rocchio [26]. He suggested the following algorithm for relevance feedback, where $\mathbf{q}_{old}$ is the original query, $\mathbf{q}_{new}$ is the revised query, $\mathbf{d}_{rel}^{(i)}$ $(1 \leq i \leq r)$ is one of the r documents the user

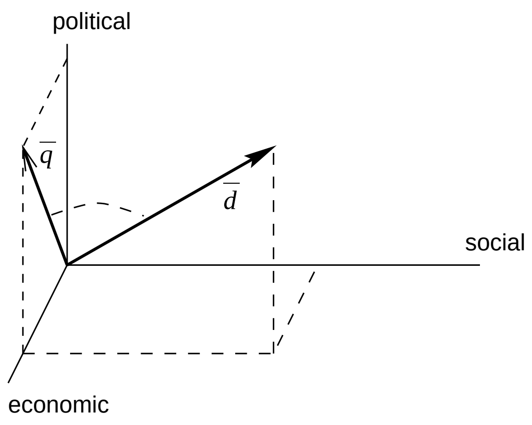


Fig. 4.2. A query and document representation in the vector space model.

selected as relevant, and $\mathbf{d}_{nonrel}^{(i)}$ $(1 \leq i \leq n)$ is one of the n documents the user selected as non-relevant:

$$\mathbf{q}_{new} \;=\; \mathbf{q}_{old} + \frac{1}{r}\sum_{i=1}^{r}\mathbf{d}_{rel}^{(i)} - \frac{1}{n}\sum_{i=1}^{n}\mathbf{d}_{nonrel}^{(i)}. \qquad (4.3)$$

The normalized vectors of documents and queries can be viewed at as points on a hypersphere at unit length from the origin. In (4.3), the first sum calculates the centroid of the points of the known relevant documents on the hypersphere. In the centroid, the angle with the known relevant documents is minimized. The second sum calculates the centroid of the points of the known non-relevant documents. Moving the query towards the centroid of the known relevant documents and away from the centroid of the known non-relevant documents is guaranteed to improve retrieval performance. The sphere is visualized for two dimensions in Figure 4.3. The figure is taken from Savino [32].

Discussion

A strong point of the vector model is the ease of explaining it to non-expert users. The main disadvantage of the vector space model is that it does not in any way subscribe what the values of the vector components should be. Now we touch the problem of term weighting which is addressed in Section 4.4. Early experiments [27] already suggested that term weighting is not a trivial problem at all. A second disadvantage of the vector space model is that it is not possible to include term dependencies into the model, for instance for modeling of phrases or adjacent terms. It is however possible to give a geometric interpretation of Boolean-structured queries, which is described in Section 4.3.3. A third problem with the vector space model is its implemen-

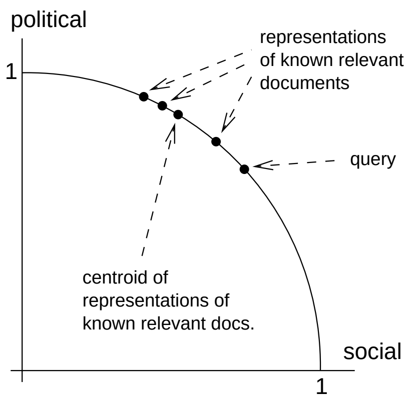

political
1
representations of known relevant documents
query
centroid of representations of known relevant docs.
social
1

Fig. 4.3. Rocchio's relevance feedback method.

tation. The calculation of the cosine measure needs the values of all vector components, which may be difficult to provide in practice [41].

4.3.2 The Probabilistic Model

Sometimes it is argued that a retrieval system should rank the documents in the collection in order of their *probability of relevance*. This seems a rather trivial requirement indeed, since the objective of information retrieval systems is defined in Chapter 1 as "to help the user to find relevant documents". However, Robertson showed that optimality of ranking by the probability of relevance can only be guaranteed if the following conditions are met. Firstly, relevance should be a dichotomous variable, either yes or no. Secondly, relevance of a document to a request should not depend on the other documents in the collection.

The Probability of Relevance

Whereas Luhn's intuitive similarity criterion raises the question "What exactly makes two representations similar?", Robertson's probability ranking principle raises the question "How, and on the basis of what data, should the probability of relevance be estimated?"

Relevance is ultimately determined by the end user. So, the probabilistic model that is based on relevance, is only useful if the system has information about relevance of documents. This information may be given by the end user as a result of relevance feedback.

It is possible that the similarity criterion and the relevance criterion do not coincide as the following reasoning shows. First let us make the notion "probability of relevance" explicit. Robertson adopted the Boolean model's viewpoint by looking at a term as a definition of a set of documents. Suppose a user enters a query containing a single term, for instance the term `social`. If all documents that fulfill the user's need were known, it would be possible to divide the document collection into four non-overlapping subsets as visualized in the Venn diagram of Figure 4.4. The figure contains additional information about the size of each of the non-overlapping subsets. The collection in question has 10,000 documents, of which 1000 contain the word "social"; only 11 documents are relevant to the query of which 1 contains the word "social". If a document is taken at random from the set of documents that are indexed with `social`, then the probability of picking a relevant document is $1/1000 = 0.0010$. If a document is taken at random from the set of documents that are *not* indexed with `social`, then the probability of relevance is bigger: $10/9000 = 0.0011$. Since the user entered only one index term, the system has only two options: either the documents indexed with the term are presented first in the ranking, or the documents that are not indexed with the term are presented first. In the example of Figure 4.4, it is wise to present the user first with documents that are not indexed with the query term `social`, that is, to present first the documents that are "dissimilar" to the query. Clearly, such a strategy violates Luhn's similarity criterion.

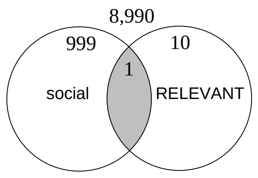


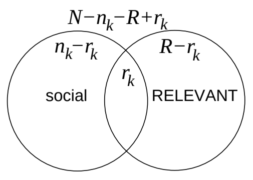
Fig. 4.4. Venn diagram of the collection given the query term `social`.

Notation

Let L be the random variable "document is relevant" with a binary sample space $\{0, 1\}$, 1 indicating relevance and 0 non-relevance. Let a query contain n terms. To each document n random variables are assigned and let D_k ($1 \leq k \leq n$) be a random variable indicating "the document belongs to the subset indexed with the kth query term" with a binary sample space $\{0, 1\}$. We concentrate on the kth query term and assume it to be `social`. We compute the four conditional probabilities $P(D_k|L)$ by the sizes of non overlapping subsets caused by the term `social`. Figure 4.5 shows the Venn diagram of documents indexed with `social`. The sizes are defined by R: the number of relevant documents, n_k: the number of documents indexed with `social`, r_k: the number of relevant documents that is indexed with `social` and N: the total number of documents in the collection.

The Binary Independence Assumption

If the user enters two terms, for instance the terms `social` and `political`, then there are four sets that must find their place in the final ranking:



$$P(D_k{=}1\,|\,L{=}1) = r_k/R$$
$$P(D_k{=}1\,|\,L{=}0) = n_k-r_k/N-R$$
$$P(D_k{=}0\,|\,L{=}1) = R-r_k/R$$
$$P(D_k{=}0\,|\,L{=}0) = N-n_k-R+r_k/N-R$$

Fig. 4.5. Definition of probabilities.

`social AND political`, `social NOT political`, `political NOT social` and `NOT(social OR political)`. Each of these Boolean subsets can be represented by a pair of binary values, the first value indicating whether the subset includes documents indexed with `social`, the second value indicating whether the subset includes documents indexed with `political`. The four Boolean subsets are represented by respectively $(1,1)$, $(1,0)$, $(0,1)$ and $(0,0)$. Below we detail the probability computations; first we consider increasing the number of query terms.

The number of non-overlapping subsets increases exponentially with the number of query terms. To make the computation of the probability of relevance possible in reasonable time, the binary independence assumption is introduced:

In documents terms occur independently from each other.

In our example this means that the probability that a relevant document contains both `social` and `political` is equal to the product of the probabilities of the terms alone:

$$\mathrm{P}(\texttt{social},\texttt{political}\,|\,L{=}1) \;=\; P(\texttt{social}\,|\,L{=}1)\; P(\texttt{political}\,|\,L{=}1).$$

We would like to compute the probability that a document is relevant given values for the random variables $D_1, D_2, \cdots, D_n$. We will show that in that computation the independence assumption will be used. First we remark that the computation may involve many multiplications of sometimes small probabilities. To prevent computational problems often a logistic transformation of probabilities is used. Equation (4.4) is a variation of Bayes' rule that uses a logistic transformation of probabilities, which is defined by $\operatorname{logit} P(L) = \log(P(L)\,/\,(1{-}P(L)))$. The transformation is strictly monotonic, so ranking documents by (4.4) will in fact rank them by the probability of relevance. Let L and D_k $(1 \leq k \leq n)$ be defined as before. Given a query of length n documents will be assigned the value defined by (4.5). Documents with the same values for $D_1, D_2, \cdots, D_n$ should be ranked equally [25, 39]. Note that duplicate query terms retrieve the same subset of documents and should be ignored in the formulas:

$$\begin{aligned}\operatorname{logit} P(L{=}1|D_1,\cdots,D_n) &= \log \frac{P(L{=}1|D_1,\cdots,D_n)}{P(L{=}0|D_1,\cdots,D_n)} \qquad (4.4)\\ &= \log \frac{P(L{=}1)P(D_1,\cdots,D_n|L{=}1)\,/\,P(D_1,\cdots,D_n)}{P(L{=}0)P(D_1,\cdots,D_n|L{=}0)\,/\,P(D_1,\cdots,D_n)}\\ &= \log \frac{P(D_1,\cdots,D_n|L{=}1)}{P(D_1,\cdots,D_n|L{=}0)} + \operatorname{logit} P(L{=}1)\\ &= \sum_{k=1}^{n} \log \frac{P(D_k|L{=}1)}{P(D_k|L{=}0)} + \operatorname{logit} P(L{=}1), \qquad (4.5)\end{aligned}$$

and, regarding the latter term,

$$\operatorname{logit} P(L{=}1) = \log \frac{P(L{=}1)}{1 - P(L{=}1)} = \log \frac{P(L{=}1)}{P(L{=}0)}.$$

The binary independence assumption is used to derive 4.5 from 4.4.

Implementation
Equation (4.5) needs some computation for subsets for which $D_k = 0$, that is for non-matching query terms. In the vector space model non-matching terms are assigned zero weight, which is usually convenient for implementation reasons. Therefore, $\sum_{k=1}^{n} \log(P(D_k{=}0|L{=}1)\,/\,P(D_k{=}0|L{=}0))$ is subtracted from the score of each document subset. This does not affect the ranking of the documents and assigns a score of zero to documents with no matching terms:

$$P(L{=}1|D_1,\cdots,D_n) \propto \sum_{\substack{k\,\in\,\text{match-}\\ \text{ing terms}}} \log \frac{P(D_k{=}1|L{=}1)\,P(D_k{=}0|L{=}0)}{P(D_k{=}1|L{=}0)\,P(D_k{=}0|L{=}1)}. \qquad (4.6)$$

Relevance Computation
The values of n_k and N are available to the system, but the values of r_k and R are only available if the user provides those to the system, typically by marking some previously retrieved documents as relevant. If r_k and R are not available to the system, it is necessary to make some assumptions about them. Robertson et al. [25] simply add 0.5 to each non-overlapping subset and Croft et al. [5] assume a constant value for $P(D_k|L{=}1)$. If the additional assumption is made that the number of relevant documents is much smaller than the size of the collection, more specifically: $R, r_k \ll N, n_k$, then documents might be ranked by a idf-like measure: $\log((N{-}n_k)\,/\,n_k)$ (see Section 4.4).

Discussion
The probabilistic model is one of the few retrieval models that do not need an additional term weighting algorithm to be implemented (see Section 4.4).

Ranking algorithms are completely derived from theory. The probabilistic model has been one of the most influential retrieval models for this very reason. Unfortunately, in many applications the distribution of terms over relevant and non-relevant documents will not be available. In these situations probability of relevance estimation is of theoretical interest only.

The main disadvantage of the probabilistic model is that it only defines a partial ranking of the documents. For short queries, the number of different subsets will be relatively low. By looking at a term as a definition of a set of documents, the probabilistic model ignores the distribution of terms within documents. In fact, one might argue that the probabilistic model suffers partially from the same defect as the Boolean model. It does not allow the user to really control the retrieved set of documents. For short queries it will sometimes assign the same rank to, for instance, the first 100 documents retrieved.

4.3.3 The *p*-norm Extended Boolean Model

The *p*-norm extended Boolean model was developed by [29], following the vector space model's metaphor of documents in a multi-dimensional Euclidean space. If the two terms `social` and `political` are again considered, the vector space spanned by the terms can be easily visualized. If document vectors are normalized to unit length, then the point (1,1) in the space represents the situation that both terms are present with weight 1 (which implies a length greater than one!). This is the desirable location for a document matching the query `social AND political`. For the query `social OR political` on the other hand, the point (0,0) representing the situation that both terms are absent, is the undesirable location for a document.

Therefore, AND-queries should rank documents in order of increasing distance from the point (1,1) and OR-queries in order of decreasing distance from the point (0,0). If the distances are properly normalized to fall between 0 and 1, then the following formulas apply. In the formula d_a denotes the weight of the term a in a document with index representation $\mathbf{d}$:

$$\begin{aligned}\text{score}(\mathbf{d}, a \text{ OR } b) &= \sqrt{\frac{(d_a - 0)^2 + (d_b - 0)^2}{2}} \\ \text{score}(\mathbf{d}, a \text{ AND } b) &= 1 - \sqrt{\frac{(1 - d_a)^2 + (1 - d_b)^2}{2}}.\end{aligned} \tag{4.7}$$

Salton [29] suggested two generalizations of the basic idea. First of all, query term weights were included to reflect the importance of individual terms. Secondly, the Euclidean distance measures were generalized by introducing a parameter p for each set operator. The resulting p-norm model uses the following formulas:

$$\begin{aligned}
\text{score}(\mathbf{d}, \mathbf{q}_{\text{OR}(p)}) &= \left(\frac{\sum_{k=1}^{m} (q_k)^p (d_k)^p}{\sum_{k=1}^{m} (q_k)^p} \right)^{1/p} \\
\text{score}(\mathbf{d}, \mathbf{q}_{\text{AND}(p)}) &= 1 - \left(\frac{\sum_{k=1}^{m} (q_k)^p (1 - d_k)^p}{\sum_{k=1}^{m} (q_k)^p} \right)^{1/p} .
\end{aligned} \tag{4.8}$$

For $p = 2$ the formulas will use the Euclidean distance measures as in (4.7). For $p = 1$ the OR-operator and the AND-operator produce the exact same results and the model behaves like the vector space model. If $p \to \infty$ then the ranking is evaluated according to the standard fuzzy set operators [44].

The p-norm model belongs to the best performing extended Boolean models. Based on recent publications about such models, the p-norm model is probably more popular for extended Boolean retrieval than other well-performing algorithms. Greiff et al. [8] copied the behavior of the p-norm model in their inference network architecture and Losada and Barreiro [15] propose a belief revision operator that is equivalent to a p-norm case.

A disadvantage of the p-norm model is that it needs an additional term weighting algorithm to be implemented.

4.3.4 Bayesian Network Models

A Bayesian network is an acyclic directed graph that encodes probabilistic dependency relationships between random variables. A directed graph is acyclic if there is no directed path $A \to \cdots \to Z$ such that $A = Z$. Probability theory ensures that the system as a whole is consistent. Some alternative names for Bayesian networks are belief networks, probabilistic independence networks, influence diagrams and causal nets [21]. This is further explained by the following simple model suggested by Turtle [37, 38], and Ribeiro [24].

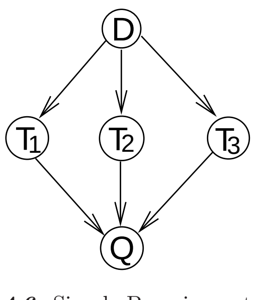


Fig. 4.6. Simple Bayesian network.

The nodes in the Bayesian network of Figure 4.6 represent binary random variables with values $\{0, 1\}$. Arrows indicate probabilistic dependency relationships, e.g., the arrow from node D to node T_1 indicates that the value for D influences the probability distribution of T_1. A missing arrow indicates probabilistic independence. So, T_1 and T_2 are independent. The random variables T_1, T_2, and T_3 stand for query terms, in our case `social`, `political`, and `economic`. If the document is relevant ($D = 1$), then the probability will be high that some of the terms will be present in the document. The information need of the user is indicated by Q. Expression $Q = 1$ indicates that the need is satisfied. The occurrence of (some of) the terms in the document will increase the probability that the information need is satisfied.

We now consider the joint probability distribution of the random variables of the Bayesian network. By the chain rule of probability, the joint probability is:

$$\begin{aligned} P(D,T_1,T_2,T_3,Q) \; &= \\ & P(D)P(T_1|D)P(T_2|D,T_1)P(T_3|D,T_1,T_2)P(Q|D,T_1,T_2,T_3). \end{aligned} \tag{4.9}$$

Independence relationships in the Bayesian network are used to simplify the joint probability distribution as follows. The second, third and fourth term in (4.10) are simplified because T_1, T_2 and T_3 are independent given their parent D. The last term is simplified because Q is independent of D given its parents T_1, T_2 and T_3:

$$P(D,T_1,T_2,T_3,Q) \; = \; P(D)\, P(T_1|D)\, P(T_2|D)\, P(T_3|D)\, P(Q|T_1,T_2,T_3). \tag{4.10}$$

We proceed using the network. If it is hypothesized that the document is relevant ($D = 1$), the probability of query fulfillment $P(Q{=}1|D{=}1)$ can be used as a score to rank the documents:

$$P(Q{=}1|D{=}1) = \frac{P(Q{=}1, D{=}1)}{P(D{=}1)} \tag{4.11}$$

$$= \frac{\sum_{t_1,t_2,t_3} P(D{=}1, T_1{=}t_1, T_2{=}t_2, T_3{=}t_3, Q{=}1)}{P(D{=}1)}. \tag{4.12}$$

The joint probability distribution defined by (4.10) can be used to calculate the score. The only thing that is still missing is the specification of the probabilities. These are shown in the form of tables in Fig. 4.7. For example, the conditional probability $P(Q|T_1,T_2,T_3)$ is given in the lower left table. With help of these tables, $P(Q{=}1|D{=}1)$ can be computed for each document.
The table of $P(Q|T_1,T_2,T_3)$, however, shows a potential difficulty of this approach. The number of probabilities that have to be specified for a node grows exponentially with its number of parents, so a query of n non-equal terms requires the specification of 2^{n+1} possible values of $P(Q|T_1,T_2,\cdots,T_n)$. Despite the simplifying assumptions made by the conditional independencies, the model has to make additional simplifying assumptions to make it possible

$P(D{=}0)$	$P(D{=}1)$
0.999	0.001

D	$P(T_1=0)$	$P(T_1=1)$
0	0.60	0.40
1	0.05	0.95

T_1	T_2	T_3	$P(Q{=}0)$	$P(Q{=}1)$
0	0	0	1.000	0.000
0	0	1	0.901	0.099
0	1	0	0.887	0.113
0	1	1	0.992	0.008
1	0	0	0.547	0.453
1	1	0	0.332	0.664
1	0	1	0.271	0.729
1	1	1	0.220	0.780

D	$P(T_2=0)$	$P(T_2=1)$
0	0.88	0.12
1	1.00	0.00

D	$P(T_3=0)$	$P(T_3=1)$
0	0.97	0.03
1	0.02	0.98

Fig. 4.7. Example specification of the model's parameters.

to calculate the probability in reasonable time. Turtle [37, page 53] therefore suggests the use of four canonical forms of $P(Q|T_1, T_2, \cdots, T_n)$ which can be computed on the fly in linear time. The four canonical forms which are called "and", "or", "sum" and "weighted sum" ("wsum" for short), are displayed in Figure 4.8. The weights w_1, w_2 and w_3 in the last columns are restricted to positive values and should sum up to one.[2]

			$P_{\text{and}}(Q)$		$P_{\text{or}}(Q)$		$P_{\text{sum}}(Q)$		$P_{\text{wsum}}(Q)$	
T_1	T_2	T_3	0	1	0	1	0	1	0	1
0	0	0	1	0	1	0	1	0	1	0
0	0	1	1	0	0	1	$\frac{2}{3}$	$\frac{1}{3}$	$1-w_3$	w_3
0	1	0	1	0	0	1	$\frac{2}{3}$	$\frac{1}{3}$	$1-w_2$	w_2
0	1	1	1	0	0	1	$\frac{1}{3}$	$\frac{2}{3}$	$1-w_2-w_3$	w_2+w_3
1	0	0	1	0	0	1	$\frac{2}{3}$	$\frac{1}{3}$	$1-w_1$	w_1
1	0	1	1	0	0	1	$\frac{1}{3}$	$\frac{2}{3}$	$1-w_1-w_3$	w_1+w_3
1	1	0	1	0	0	1	$\frac{1}{3}$	$\frac{2}{3}$	$1-w_1-w_2$	w_1+w_2
1	1	1	0	1	0	1	0	1	0	1

Fig. 4.8. Canonical forms of $P(Q|T_1, T_2, T_3)$.

Suppose for now that the values of $P(T_1|D)$, $P(T_2|D)$ and $P(T_3|D)$ are known and given by p_1, p_2 and p_3. The calculation of $P(Q{=}1|D{=}1)$ by the canonical forms of Figure 4.8 will give the same results as the following calculations, which only require linear time:

[2] The definition of "wsum" by Turtle [37] is more general.

$$\begin{aligned}
P_{\text{and}}(Q{=}1|D{=}1) &= p_1\, p_2\, p_3 \\
P_{\text{or}}(Q{=}1|D{=}1) &= 1 - (1{-}p_1)(1{-}p_2)(1{-}p_3) \\
P_{\text{sum}}(Q{=}1|D{=}1) &= (p_1 + p_2 + p_3)\,/\,3 \\
P_{\text{wsum}}(Q{=}1|D{=}1) &= w_1\, p_1 + w_2\, p_2 + w_3\, p_3.
\end{aligned} \tag{4.13}$$

The main advantage of the Bayesian network models [38] is that the network topology can be used to combine evidence in a complex way. Many other recent approaches to information retrieval seek for new ways of combining evidence from multiple sources [7, 33, 40, 43].

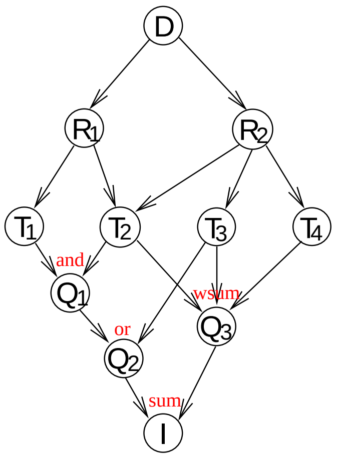


Fig. 4.9. Complex Bayesian network.

Figure 4.9 shows such a complex Bayesian network. In the network R_1 and R_2 define different representations of the document, for instance one might represent the document's title words, whereas the other might represent words from the abstract. The model's probabilities might indicate that title words are more important than words from the abstract. The nodes Q_1, Q_2 and Q_3 represent different queries for the same information need, which is represented by the node I. The query represented by Q_2 is evaluated as `or(and(`T_1 T_2`)` T_3`))`, whereas the query Q_3 is evaluated as `wsum(`T_2 T_3 T_4`)`.

There are two disadvantages of the Bayesian network models presented in this section. Firstly, the approaches do not suggest how the probability measures $P(T_i|D)$, $(1 \leq i \leq n)$ should be estimated. Instead, the approaches suggest the use of Bayesian probabilities. In a nutshell, the Bayesian probability of an event is a person's degree of belief in that event, which does not have to refer to a physical mechanism or experiment. In contrast, the classical probability always implies such an experiment and therefore can always be interpreted as a relative frequency. Considering probabilities as a person's degree of belief is quite practical if a medical expert system is built [9]. For full text information retrieval systems however, experts are by definition not available for specifying the probabilities of the network because it implies manual indexing of the collection. The models therefore use one of the term weighting algorithms that use term frequencies and document frequencies as presented in Section 4.4. The joint probability distribution defined by (4.10) can be used as follows to calculate the score.

A second disadvantage of the Bayesian network models presented in this section is that the calculation of the probabilities generally takes exponential time in the number n of non-equal query terms. The introduction of the four canonical forms solves this problem, but it could have been solved by the network topology. For instance the definition of P_{and} in (4.13) actually suggests (conditional) independence between the probabilities p_1, p_2 and p_3 and, for instance the definition of P_{wsum} suggests the use of a mixture model topology [13]. By using the four canonical forms, the network is tractable if it is used for inference, but it is still intractable if used for updating the probabilities. Updating the probabilities might be an effective approach to relevance feedback. Although the Bayesian network formalism comes with efficient learning algorithms, these algorithms can in practice not be applied in reasonable time on the network model presented in this section [36].

4.3.5 Language Model

A language model is a mathematical model of a language. Such a model can be very simple, for instance a list with the words of a language together with the frequency with which the word occurs in sentences. Language models have been around for quite a long time. They were first applied by Andrei Markov at the beginning of the 20th century to model letter sequences in works of Russian literature [17]. Later on language models were also used to model word sequences [34]. At the end of the 1970s language models were first successfully used for automatic speech recognition. Recently, statistical language models are very popular in the area of information retrieval. In this case one builds a simple language model for each document in the collection and given a query, documents are ranked by the probability that the language model of each document generated the query. It may be instructive to describe the process of generating the query from the model as if it were a physical process.

An Informal Description: the Urn Model Metaphor

The metaphor of "urn models" [19] might give more insight. Instead of drawing balls at random with replacement from an urn, we will consider the process of drawing words at random with replacement from a document. Suppose someone selects one document in the document collection; draws at random, one at a time, with replacement, ten words from this document and hands those ten words (the query terms) over to the system. The system can now make an educated guess as from which document the words came from, by calculating for each document the probability that the ten words were sampled from it and by ranking the documents accordingly. The intuition behind it is that users have a reasonable idea of which terms are likely to occur in documents of interest and will choose query terms accordingly [22]. In practice, some query terms do not occur in any of the relevant documents. This can be modeled by a slightly more complicated urn model. In this case the person that draws at random the ten words, first decides for each draw if he will draw randomly from a relevant document or randomly from the entire collection. The yes/no decision of drawing from a relevant document or not, will also be assigned a probability. This probability will be called the relevance weight of a term, because it defines the distribution of the term over relevant and non-relevant documents. For *ad hoc* retrieval all non-stop-words in the query will be assigned the same relevance weight. The user's feedback might be used to re-estimate the relevance weight for each query term.

Definition of the Corresponding Probability Measures

Based on the ideas mentioned above, probability measures can be defined to rank the documents given a query. The probability that a query $T_1, T_2, \cdots, T_n$ of length n is sampled from a document with document identifier D is defined by:

$$P(T_1, T_2, \cdots, T_n|D) = \prod_{i=1}^{n}((1-\lambda_i)P(T_i) + \lambda_i P(T_i|D)). \qquad (4.14)$$

In the formula, $P(T)$ is the probability of drawing a term randomly from the collection, $P(T|D)$ is the probability of drawing a term randomly from a document and λ_i is the relevance weight of the term. If a query term is assigned a relevance weight of $\lambda_i = 1$, then the term is treated as in exact matching: the system will assign zero probability to documents in which the term does *not* occur. If a query term is assigned a relevance weight of 0, then the term is treated like a stop word: the term does not have any influence on the final ranking. It can be shown that this probability measure can be rewritten to a *tf.idf* term weighting algorithm.

Parameter Estimation

It is good practice in information retrieval to use the term frequency and document frequency as the main components of term weighting algorithms. The term frequency $\mathrm{tf}(t, d)$ is defined by the number of times the term t occurs in the document d. The document frequency $df(t)$ is defined by the number of documents in which the term t occurs. Estimation of $P(T)$ and $P(T|D)$ in (4.14) might therefore be done as follows [10, 12]:

$$P(T_i = t_i | D = d) = \frac{\mathrm{tf}(t_i, d)}{\sum_t \mathrm{tf}(t, d)} \tag{4.15}$$

$$P(T_i = t_i) = \frac{df(t_i)}{\sum_t df(t)}. \tag{4.16}$$

From the viewpoint of using language models for retrieval and from the viewpoint of the urn model metaphor, (4.16) would not be the obvious method for the estimation of $P(T)$. One might even argue that it violates the axioms of probability theory, because $P(T_i{=}t_{i1} \cup T_i{=}t_{i2}) \neq P(T_i{=}t_{i1}) + P(T_i{=}t_{i2})$ if t_{i1} and t_{i2} co-occur in some documents. Therefore the following equation, (4.16b), would be the preferred method for the estimation of $P(T)$, where the collection frequency $\mathrm{cf}(t)$ is defined by the number of times the term t occurs in the entire collection:

$$P(T_i = t_i) \quad = \quad \frac{\mathrm{cf}(t_i)}{\sum_t \mathrm{cf}(t)} \quad = \quad \frac{\sum_d \mathrm{tf}(t_i, d)}{\sum_d \sum_t \mathrm{tf}(t, d)} \;. \tag{4.16b}$$

The latter method was used by various authors [1, 18, 20, 22]. We try to relate the language modeling approach to the traditional approaches, so we will use former method. By using (4.16), the language modeling approach to information retrieval gives a strong theoretical backup for using *tf.idf* term weighting algorithms: a backup that is not provided by the traditional retrieval models. The prior probability $P(D{=}d)$ that a document d is relevant, might assumed to be uniformly distributed, in which case the formulas above suffice. Alternatively, it might be assumed that the prior probability of a document being relevant is proportional to the length of the document as in:

$$P(D = d) = \frac{\sum_t \mathrm{tf}(t, d)}{\sum_t \sum_d \mathrm{tf}(t, d)} \tag{4.17}$$

It can be included in the final ranking algorithm by adding the logarithm of (4.17) to the document scores as a final step.

4.3.6 Ranking in Google

The World-Wide Web has become increasingly important. A Web page may contain all kinds of information: text, images, and so on. We consider a Web page to be a hypertext document, which means that besides text also the HTML referencing mechanism (=link) is available. Many search engines have been developed to address the huge document collection offered by the Web; the dominant search engine is without doubt Google. This engine offers high performance and ease of use. Ease of use is achieved by allowing the user to issue natural language search terms that are subsequently processed as if they were separated by ANDs. In the meantime other Boolean operators (OR, NOT) are allowed via an advanced search interface. Actually, Google uses a Boolean matching model.

The decisive factor, however, in the success of Google is probably its ranking mechanism. An important part in this mechanism is the so-called PageRank algorithm [3]. The algorithm is based on the citation index, which is generally accepted in the academic world: the importance of a paper can be judged by the number of references to it. In Web terms: the number of links referring to a Web page.

Within the past few years many adjustments and modifications regarding Google's ranking mechanism have occurred. PageRank is only part of the mechanism determining what results get displayed high up in a Google output. For example, there is some evidence to suggest that Google is paying a lot of attention these days to the text in a link's anchor when deciding the relevance of a target page. PageRank remains, however, an interesting algorithm. Below we will describe one of the first versions of the algorithm.

PageRank

The original PageRank algorithm as described by Brin and Page is given by:

$$PR(A) = (1-d) + d\left(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right), \tag{4.18}$$

where $PR(A)$ is the PageRank of page A, $PR(T_i)$ is the PageRank of a page T_i, $C(T_i)$ is the number of outgoing links (=references) from the page T_i, and d is a damping factor in the range $0 < d < 1$, usually $d = 0.85$.

So, the PageRank of a Web page is the sum of the PageRanks of all pages referring to the page (its incoming links), divided by the number of links on each of those pages (its outgoing links). This means that the influence of a referring page is positively related to its PageRank and negatively by the number of references it makes.

PR is a recursive function: To compute $PR(A)$, we must know $PR(T_i)$ of all pages referring to A. But to compute these PR's we need to know the PR of referring pages (may be from page A), and so on. An iterative algorithm solves this problem after assigning a starting PageRank value of 1 to each page on the web.

Example

Consider a small Web consisting of three pages A, B, and C. Page A refers to B and C; page B to A and C, and page C to B. Let us assume that the damping factor $d = 0.5$. Actually, the damping factor appears to have a significant influence on the convergence characteristics of the algorithm, but for explanation purposes it is not relevant. Now we get:

$$\begin{aligned}
PR(A) &= 0.5 + 0.5\left(\frac{PR(B)}{2}\right)\\
PR(B) &= 0.5 + 0.5\left(\frac{PR(A)}{2} + PR(C)\right)\\
PR(C) &= 0.5 + 0.5\left(\frac{PR(A)}{2} + \frac{PR(B)}{2}\right).
\end{aligned}$$

Solving these equations results in the following PageRank values for the pages:

$$PR(A) = \frac{4}{5}; \quad PR(B) = \frac{6}{5}; \quad PR(C) = 1.$$

The sum of all PR's equals the total number of Web pages ($= 3$).

Page and Brin published a second version of the algorithm to compute PR. In that version the term $1-d$ is divided by N, the total number of pages on the web. It can be shown that the PageRanks now form a probability distribution over the web pages, so they sum up to one.

Random Surfer Model

In their publications, Page and Brin describe the random server model which is a justification for the PageRank formula. The random server visits a Web page with a probability which is derived from the PR of the page. Now the surfer randomly selects one link on the page and follows that link. However, from time to time (probability $(1-d)/N$) the surfer gets bored, does not follow a selected link, but jumps to another random page instead. In this way an intuitive reasoning for the computation of PR results.

Remark

It may be commercially interesting for a Web page to get a high ranking. So Google pays a lot of attention to obstruct efforts of Webmasters to elude the system. For instance, Google introduced the concept of the “importance” of the page. This may help to minimize the effect of artificially generated Web pages which refer to a certain page in order to increase its ranking.

4.4 Term Weighting

Of the models presented in Section 4.3, the vector space model, the p-norm model and the Bayesian network models all need an additional term weighting algorithm before they can be implemented. Weighting of search terms is the single most important factor in the performance of information retrieval systems. The development of term weighting approaches is as much an art as it is a science: Literally thousands of term weighting algorithms were used experimentally during the last 25 years, especially within the Smart projects.

These algorithms often imply the use of some statistics on the terms, that is, they somehow take into account the number of occurrences of terms in the documents or in the index to compute rankings.

In this section we give an example of term weighting and choose the tf.idf weights of the original Smart retrieval system. This system was developed at Harvard University in the early 1960s and later developed at Cornell University. Salton [31] experimented with weighting algorithms that use the so-called *inverse document frequency*. They suggested to combine it with the frequency of a term within a document, the *term frequency*, tf for short. The assumption is that if a term occurs frequently, it is likely to be characteristic for a document.

The document frequency df of a term is defined by the number of documents a term occurs in. A term with a low document frequency is more specific than a term with a high document frequency. Sparck et al. [35] suggested that therefore, the system should treat matches on non-frequent terms as more valuable than ones on frequent terms. An intuitive way to relate the matching value of a term to its document frequency is suggested by a Zipf-like distribution of words in a vocabulary [17]. If $f(df) = m$ such that $2^{m-1} < df \leq 2^m$, and N is the number of documents in the collection, then the weight of a term that occurs df times is $f(N) - f(df)$.[3] A continuous approximation of f is the logarithm to the base 2. The ranking algorithm is displayed in Figure 4.10. The weight $\log(N/df)$ will be called the "inverse document frequency": *idf* for short.

$$\text{cosine:} \quad \text{score}(\mathbf{d}, \mathbf{q}) = \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2} \cdot \sqrt{\sum_{k=1}^{m} (q_k)^2}}$$

$$\text{term weights:} \quad d_k = q_k = \text{tf} \cdot \log \frac{N}{df}$$

Fig. 4.10. Original *tf.idf* with cosine normalization (tfc.tfc).

The introduction of the so-called tf.idf weights is one of the major breakthroughs of term weighting in information retrieval. Most modern weighting algorithms are versions of the family of tf.idf weighting algorithms. Salton's original tf.idf weights perform relatively poor, in some cases even poorer than simple idf weighting.

Salton [28] summarizes the results of 20 years of research into term weighting with the Smart system. A total of 1800 different combinations of term weight assignments were used experimentally, of which 287 were found to be

[3] The adding of 1 used by Sparck–Jones [35] was ignored because it is no longer used in later papers.

distinct. Experimental results of these term weighting algorithms on six document collections were reported. Term weighting algorithms were named by three letter combinations. The first letter indicated the tf component, the second component indicates the idf component and the third component indicates the normalization component. For instance, the three letter code tfc is the code for the original tf.idf weights with cosine normalization introduced above. They concluded that the best performing algorithm is one that maps the document vectors differently in the vector space than the query vectors. Figure 4.11 displays the tfc.nfc formula which uses a normalized tf factor for the query term weights.

$$\begin{array}{rrcl}
\textsf{cosine:} & \text{score}(\mathbf{d},\mathbf{q}) & = & \dfrac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m}(d_k)^2} \cdot \sqrt{\sum_{k=1}^{m}(q_k)^2}} \\
\textsf{document term weight:} & d_k & = & \text{tf} \cdot \log \dfrac{N}{df} \\
\textsf{query term weight:} & q_k & = & \left(0.5 + \dfrac{0.5\,\text{tf}}{\max \text{tf}}\right) \cdot \log \dfrac{N}{df}
\end{array}$$

Fig. 4.11. tfc.nfc term weighting algorithm.

4.5 Summary

A brief description of thirty years of IR research has been given. Two classes of users are identified, namely professional indexers and searchers on one hand and casual end users on the other. The formulation of queries for these two groups is described and the process of deriving a query from a natural language request is briefly sketched. Most attention has been paid to matching problems. The Boolean approach resulted in exact matching; other approaches included ranking of resulting documents (which means inexact matching). Several approaches to ranking are given. The vector space approach maps documents and queries in a n-dimensional vector space. Relevance feedback nicely fits into this approach. The probabilistic approach tries to estimate the probability that a document is relevant for the user. If users are able to characterize relevant (or irrelevant) documents this approach may be useful. An example may be email documents where users are able to specify terms to characterize spam. The Bayesian approach is useful when evidence from many different sources have to be integrated. Think of evidence coming from an audio, and image channel in a video. A language model is originally a (simple) mathematical model of a language. In the meantime these models are heavily used in IR. Systems built on these models are very competitive and perform as well as, or better than, today's top-performing algorithms. The World-Wide Web is a collection of interconnected Web pages. Google searches the Web and its ranking algorithm (PageRank) takes these connections (=hyperlinks) into account. Some retrieval models perform dramatically better if query terms that occur in documents get a weight. Some variants have been discussed, but term frequency (how often does the term occur in a document) and document frequency (in how many documents does the term occur) play an important part.

4.6 Further Reading

This chapter briefly introduces probabilistic retrieval models. Fuhr [6] elaborates on this topic.

IR deals with documents containing free text. For presentation purposes free text is more and more embedded in a language like HTML. More recent is the development to use XML to structure documents. This poses new challenges to IR systems. For instance, how to deal with queries that contain conditions related to both structure and content? Blanken et al. [2] give more information on IR and structured data.

Retrieval models deal with structured queries, relevance feedback, ranking, term weighting, and so on. There have been attempts to model these phenomena into one framework [38]. Section 4.3.5 introduces the language model. An extension of the model integrates structured queries and relevance feedback into one mathematical framework [11].

Implementation aspects did not get much attention. Consider queries like: how to store terms in an index, and how to access indexes? Efficiency is of course an important topic. Witten et al. deal with indexing in Chapter 3 of their book [42].

In this chapter, architectural aspects are totally neglected. Brin and Page [3] give more information about architectural aspects of Google.

References

1. A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 222–229, 1999.
2. H.M. Blanken, T. Grabs, H.-J. Schek, and G. Weikum, editors. *Intelligent Search on XML data: Applications, Languages, Models, Implementations, and Benchmarks*, volume 2818. Springer: LNCS series, 2003.
3. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.
4. G.G. Chowdhury. *Introduction to modern information retrieval*. Wiley, 1998.
5. W.B. Croft and D.J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4):285–295, 1979.
6. N. Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

7. N. Fuhr. Probabilistic datalog: A logic for powerful retrieval methods. In *Proceedings of the 18th ACM Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 282–290, 1995.
8. W.R. Greiff, W.B. Croft, and H.R. Turtle. Computationally tractable probabilistic modeling of boolean operators. In *Proceedings of the 20th ACM Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 119–128, 1997.
9. D.E. Heckerman. *Probabilistic Similarity Networks*. MIT Press, 1991.
10. D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 569–584, 1998.
11. D. Hiemstra and A.P. de Vries. Relating the new language models of information retrieval to the traditional retrieval models. Technical Report TR-CTIT-00-09, Centre for Telematics and Information Technology, 2000. `http://www.ub.utwente.nl/webdocs/ctit/1/00000022.pdf`.
12. D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the seventh Text Retrieval Conference TREC-7*, pages 227–238. NIST Special Publication 500-242, 1999.
13. M.I. Jordan, editor. *Learning in Graphical Models*. Kluwer Academic Press, 1998.
14. G. Kowalski. *Information Retrieval Systems: Theory and Implementation*. Kluwer Academic Publishers, 1997.
15. D.E. Losada and A. Barreiro. Using a belief revision operator for document ranking in extended boolean models. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 66–73, 1999.
16. H.P. Luhn. A statistical approach to mechanised encoding and searching of litary information. *IBM Journal of Research and Development*, 1(4):309–317, 1957.
17. C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
18. D.R.H. Miller, T. Leek, and R.M. Schwartz. A hidden Markov model information retrieval system. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 214–221, 1999.
19. A.M. Mood and F.A. Graybill. *Introduction to the Theory of Statistics, Second edition*. McGraw-Hill, 1963.
20. K. Ng. A maximum likelihood ratio information retrieval model. In *Proceedings of the eighth Text Retrieval Conference, TREC-8*. NIST Special Publications, to appear.
21. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
22. J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 275–281, 1998.
23. M.F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.
24. B.A.N. Ribeiro and R. Muntz. A belief network model for ir. In *Proceedings of the 19th ACM Conference on Research and Development in Information Retrieval (SIGIR'96)*, pages 252–260, 1996.
25. S.E. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

26. J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.
27. G. Salton. *The SMART retrieval system: Experiments in automatic document processing*. Prentice-Hall, 1971.
28. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
29. G. Salton, E.A. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.
30. G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
31. G. Salton and C.S. Yang. On the specification of term values in automatic indexing. *Jounral of Documentation*, 29(4):351–372, 1973.
32. P. Savino and F. Sebastiani. Essential bibliography on multimedia information retrieval, categorisation and filtering. In *Slides of the 2nd European Digital Libraries Conference Tutorial on Multimedia Information Retrieval*, 1998.
33. F. Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR'94)*, pages 122–130, 1994.
34. C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.
35. K. Sparck-Jones. A statistical interpretation of term specifity and its application in retrieval. *Journal of Documentation*, 28(1):11–20, 1972.
36. H. Turtle and W.B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.
37. H.R. Turtle. *Inference Networks for Document Retrieval*. PhD thesis, Centre for Intelligent Information Retrieval, University of Massachusetts Amherst, 1991.
38. H.R. Turtle and W.B. Croft. A comparison of text retrieval models. *The Computer Journal*, 35(3):279–290, 1992.
39. C.J. van Rijsbergen. *Information Retrieval, second edition*. Butterworths, 1979. `http://www.dcs.gla.ac.uk/Keith/Preface.html`.
40. C.J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6):481–485, 1986.
41. I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, 1994.
42. I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Indexing*. Morgan Kaufmann, 1999.
43. S.K.M. Wong and Y.Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13:38–68, 1995.
44. L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.