1

Information Retrieval Models

Djoerd Hiemstra

COPYRIGHTED MATERIAL

1.1 Introduction

Many applications that handle information on the internet would be completely inadequate without the support of information retrieval technology. How would we find information on the world wide web if there were no web search engines? How would we manage our email without spam filtering? Much of the development of information retrieval technology, such as web search engines and spam filters, requires a combination of *experimentation* and *theory*. Experimentation and rigorous empirical testing are needed to keep up with increasing volumes of web pages and emails. Furthermore, experimentation and constant adaptation of technology is needed in practice to counteract the effects of people who deliberately try to manipulate the technology, such as email spammers. However, if experimentation is not guided by theory, engineering becomes trial and error. New problems and challenges for information retrieval come up constantly. They cannot possibly be solved by trial and error alone. So, what is the theory of information retrieval?

There is not one convincing answer to this question. There are many theories, here called *formal models,* and each model is helpful for the development of some information retrieval tools, but not so helpful for the development of others. In order to understand information retrieval, it is essential to learn about these retrieval models. In this chapter, some of the most important retrieval models are gathered and explained in a tutorial style. But first, we will describe what exactly it is that these models model.

1.1.1 Terminology

An information retrieval system is a software programme that stores and manages information on documents, often textual documents, but possibly multimedia. The system assists users in finding the information they need. It does not explicitly return information or answer questions. Instead, it informs on the existence and location of documents that might contain the desired information. Some suggested documents will, hopefully, satisfy the user's information need. These documents are called *relevant* documents. A perfect retrieval system would retrieve only the relevant documents and no irrelevant documents. However, perfect retrieval systems do not exist and will not exist, because search statements are necessarily incomplete and relevance depends on the subjective opinion of the user. In practice, two users may pose the same query to an information retrieval system and judge the relevance of the retrieved documents differently. Some users will like the results, others will not.

Information Retrieval: Searching in the 21st Century edited by A. Göker & J. Davies
© 2009 John Wiley & Sons, Ltd

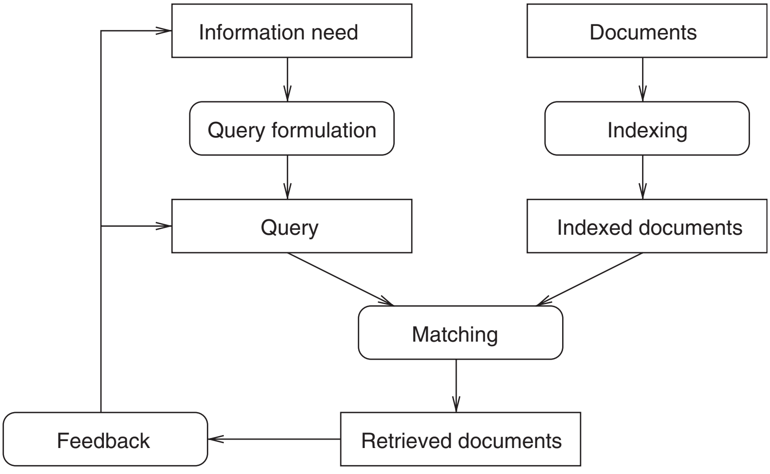

Figure 1.1 Information retrieval processes

There are three basic processes an information retrieval system has to support: the representation of the content of the documents, the representation of the user's information need, and the comparison of the two representations. The processes are visualised in Figure 1.1. In the figure, squared boxes represent data and rounded boxes represent processes.

Representing the documents is usually called the *indexing* process. The process takes place offline, that is, the end user of the information retrieval system is not directly involved. The indexing process results in a representation of the document. Often, full-text retrieval systems use a rather trivial algorithm to derive the index representations, for instance an algorithm that identifies words in an English text and puts them to lower case. The indexing process may include the actual storage of the document in the system, but often documents are only stored partly, for instance only the title and the abstract, plus information about the actual location of the document.

Users do not search just for fun, they have a need for information. The process of representing their *information need* is often referred to as the *query formulation* process. The resulting 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 the user better understanding his/her information need: This is denoted by the feedback process in Figure 1.1.

The comparison of the query against the document representations is called the *matching* process. The matching process usually results in 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 towards the top of the ranked list, minimising the time the user has to invest in reading the documents. Simple, but effective ranking algorithms use the frequency distribution of terms over documents, but also statistics over other information, such as the number of hyperlinks that point to the document. Ranking algorithms based on statistical approaches easily halve the time the user has to spend on reading documents. The theory behind ranking algorithms is a crucial part of information retrieval and the major theme of this chapter.

1.1.2 What is a model?

There are two good reasons for having models of information retrieval. The first is that models guide research and provide the means for academic discussion. The second reason is that models can serve as a blueprint to implement an actual retrieval system.

Mathematical models are used in many scientific areas with the objective to understand and reason about some behaviour or phenomenon in the real world. One might for instance think of a model of our solar system that predicts the position of the planets on a particular date, or one might think of a model of the world climate that predicts the temperature, given the atmospheric emissions of greenhouse gases. A model of information retrieval predicts and explains what a user will find relevant, given the user query. The correctness of the model's predictions can be tested in a controlled experiment. In order to do predictions and reach a better understanding of information retrieval, models should be firmly grounded in intuitions, metaphors and some branch of mathematics. Intuitions are important because they help to get a model accepted as reasonable by the research community. Metaphors are important because they help to explain the implications of a model to a bigger audience. For instance, by comparing the earth's atmosphere with a greenhouse, non-experts will understand the implications of certain models of the atmosphere. Mathematics is essential to formalise a model, to ensure consistency, and to make sure that it can be implemented in a real system. As such, a model of information retrieval serves as a blueprint which is used to implement an actual information retrieval system.

1.1.3 Outline

The following sections will describe a total of eight models of information retrieval rather extensively. Many more models have been suggested in the information retrieval literature, but the selection made in this chapter gives a comprehensive overview of the different types of modelling approaches. We start out with two models that provide structured query languages, but no means to rank the results in Section 1.2. Section 1.3 describes vector space approaches, Section 1.4 describes probabilistic approaches, and Section 1.5 concludes this chapter.

1.2 Exact Match Models

In this section, we will address two models of information retrieval that provide exact matching, i.e. documents are either retrieved or not, but the retrieved documents are not ranked.

1.2.1 The Boolean model

The Boolean model is the first model of information retrieval and probably also the most criticised model. The model 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, the logical product called AND, the logical sum called OR and the logical difference called NOT. Combining terms with the 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 the term `social` and the term `economic`, i.e. the intersection of both sets. 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 the term `social` or the term `political`, or both, i.e. the union of both sets. This is visualised in the Venn diagrams of Figure 1.2 in which each set of documents is visualised by a disc. The intersections of these discs and their complements divide the document collection into 8 non-overlapping regions, the unions of which give 256 different Boolean combinations of 'social, political and economic documents'. In Figure 1.2, the retrieved sets are visualised by the shaded areas.

An advantage of the Boolean model is that it gives (expert) users a sense of control over the system. It is immediately clear why a 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. For untrained users, the model has a number of clear disadvantages. Its main

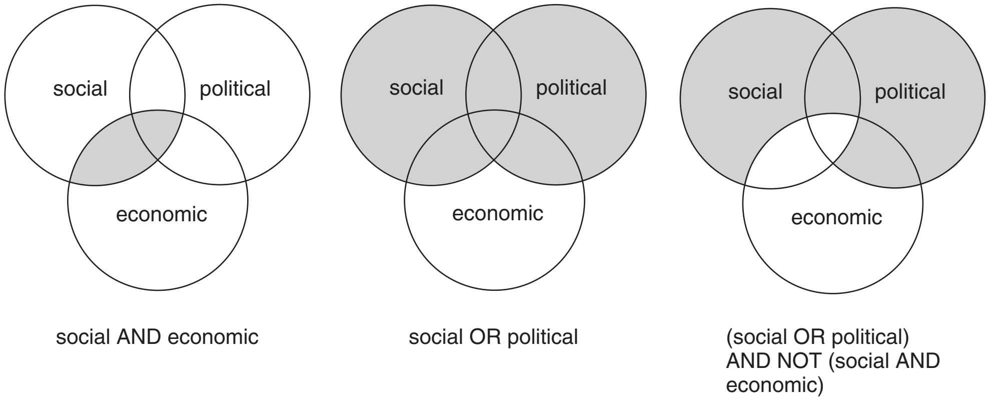

Figure 1.2 Boolean combinations of sets visualised as Venn diagrams

disadvantage is that it does not provide a ranking of retrieved documents. The model either retrieves a document or not, which might lead to the system making rather frustrating decisions. For instance, the query `social AND worker AND union` will of course not retrieve a document indexed with `party`, `birthday` and `cake`, but will likewise not retrieve a document indexed with `social` and `worker` that lacks the term `union`. Clearly, it is likely that the latter document is more useful than the former, but the model has no means to make the distinction.

1.2.2 Region models

Region models (Burkowski 1992; Clarke *et al*. 1995; Navarro and Baeza-Yates 1997; Jaakkola and Kilpelainen 1999) are extensions of the Boolean model that reason about arbitrary parts of textual data, called segments, extents or *regions*. Region models model a document collection as a linearised string of words. Any sequence of consecutive words is called a region. Regions are identified by a start position and an end position. Figure 1.3 shows a fragment from Shakespeare's *Hamlet* for which we numbered the word positions. The figure shows the region that starts at word 103 and ends at word 131. The phrase 'stand, and unfold yourself' is defined by the region that starts on position 128 in the text, and ends on position 131. Some regions might be predefined because they represent a logical element in the text, for instance the line spoken by Bernardo which is defined by region (122, 123).

Region systems are not restricted to retrieving documents. Depending on the application, we might want to search for complete plays using some textual queries, we might want to search for scenes referring to speakers, we might want to retrieve speeches by some speaker, we might want to search for single lines using quotations and referring to speakers, etc. When we think of the Boolean model as operating on sets of documents where a document is represented by a nominal identifier, we could think of a region model as operating on sets of regions, where a region is represented by two ordinal identifiers: the start position and the end position in the document collection. The Boolean operators `AND`, `OR` and `NOT` might be defined on sets of regions in a straightforward way as set intersection, set union and set complement. Region models use at least two more operators: `CONTAINING` and `CONTAINED_BY`. Systems that supports region queries can process complex queries, such as the following that retrieves all lines in which Hamlet says 'farewell': `(<LINE> CONTAINING farewell) CONTAINED_BY (<SPEECH> CONTAINING (<SPEAKER> CONTAINING Hamlet))`.

There are several proposals of region models that differ slightly. For instance, the model proposed by Burkowski (1992) implicitly distinguishes mark-up from content. As above, the query `<SPEECH> CONTAINING Hamlet` retrieves all speeches that contain the word 'Hamlet'. In later publications Clarke *et al*. (1995) and Jaakkola and Kilpelainen (1999) describe region models that do *not* distinguish

⋮
<ACT>
<TITLE>ACT^{103} I^{104}</TITLE>
<SCENE>
<TITLE>SCENE^{105} I.^{106} Elsinore.^{107} A^{108} platform^{109} before^{110} the^{111} castle.^{112}</TITLE>
<STGDIR>FRANCISCO^{113} at^{114} his^{115} post.^{116} Enter^{117} to^{118} him^{119} BERNARDO^{120}</STGDIR>
<SPEECH>
<SPEAKER>BERNARDO^{121}</SPEAKER>
<LINE>Who's^{122} there?^{123}</LINE>
</SPEECH>
<SPEECH>
<SPEAKER>FRANCISCO^{124}</SPEAKER>
<LINE>Nay,^{125} answer^{126} me:^{127} stand,^{128} and^{129} unfold^{130} yourself.^{131}</LINE>
⋮

Figure 1.3 Position numbering of example data

mark-up from content. In their system, the operator `FOLLOWED_BY` is needed to match opening and closing tags, so the query would be somewhat more verbose: `(<speech> FOLLOWED_BY </speech>) CONTAINING Hamlet`. In some region models, such as the model by Clarke *et al*. (1995), the query `A AND B` does not retrieve the intersection of sets A and B, but instead retrieves the smallest regions that contain a region from both set A and set B.

1.2.3 Discussion

The Boolean model is firmly grounded in mathematics and its intuitive use of document sets provides a powerful way of reasoning about information retrieval. The main disadvantage of the Boolean model and the region models is their inability to rank documents. For most retrieval applications, ranking is of the utmost importance and ranking extensions have been proposed of the Boolean model (Salton *et al*. 1983) as well as of region models (Mihajlovic 2006). These extensions are based on models that take the need for ranking as their starting point. The remaining sections of this chapter discuss these models of ranked retrieval.

1.3 Vector Space Approaches

Hans Peter Luhn was the first to suggest a statistical approach to searching information (Luhn 1957). He suggested that in order to search a document collection, the user should first prepare a document that is similar to the documents needed. The degree of similarity between the representation of the prepared document and the representations of the documents in the collection is used to rank the search results. Luhn formulated his similarity criterion as follows:

> The more two representations agreed in given elements and their distribution, the higher would be the probability of their representing similar information.

Following Luhn's similarity criterion, a promising first step is to count the number of elements that the query and the index representation of the document share. If the document's index representation is a vector $\vec{d} = (d_1, d_2, \cdots, d_m)$ of which each component $d_k (1 \leq k \leq m)$ is associated with an index

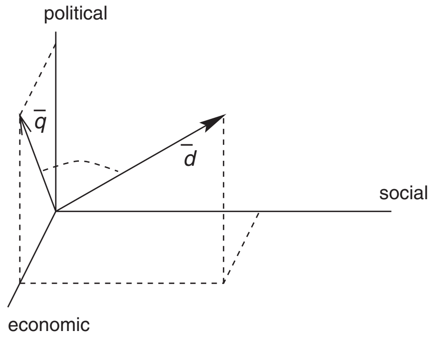

Figure 1.4 A query and document representation in the vector space model

term; and if the query is a similar vector $\vec{q} = (q_1, q_2, \cdots, q_m)$ of which the components are associated with the same terms, then a straightforward similarity measure is the vector inner product:

$$\text{score}(\vec{d}, \vec{q}) = \sum_{k=1}^{m} d_k \cdot q_k \tag{1.1}$$

If the vector has binary components, i.e. the value of the component is 1 if the term occurs in the document or query and 0 if not, then the vector product measures the number of shared terms. A more general representation would use natural numbers or real numbers for the components of the vectors $\vec{d}$ and $\vec{q}$.

1.3.1 The vector space model

Gerard Salton and his colleagues suggested a model based on Luhn's similarity criterion that has a stronger theoretical motivation (Salton and McGill 1983). They considered the index representations and the query as vectors embedded in a high-dimensional Euclidean space, where each term is assigned a separate dimension. The similarity measure is usually the cosine of the angle that separates the two vectors $\vec{d}$ and $\vec{q}$. The cosine of an angle is 0 if the vectors are orthogonal in the multidimensional space and 1 if the angle is 0°. The cosine formula is given by:

$$\text{score}(\vec{d}, \vec{q}) = \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2} \cdot \sqrt{\sum_{k=1}^{m} (q_k)^2}} \tag{1.2}$$

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 visualise the document and query vectors. Figure 1.4 visualises an example document vector and an example query vector in the space that is spanned by the three terms `social`, `economic` and `political`. The intuitive geometric interpretation makes it relatively easy to apply the model to new information retrieval problems. The vector space model guided research in, for instance, automatic text categorisation and document clustering.

Measuring the cosine of the angle between vectors is equivalent to normalising the vectors to unit length and taking the vector inner product. If index representations and queries are properly normalised, then the vector product measure of Equation (1.1) does have a strong theoretical motivation. The formula then becomes:

$$\text{score}(\vec{d}, \vec{q}) = \sum_{k=1}^{m} n(d_k) \cdot n(q_k)$$

where

$$n(v_k) = \frac{v_k}{\sqrt{\sum_{k=1}^{m}(v_k)^2}} \tag{1.3}$$

1.3.2 Positioning the query in vector space

Some rather *ad hoc*, but quite successful retrieval algorithms are nicely grounded in the vector space model if the vector lengths are normalised. An example is the relevance feedback algorithm by Joseph Rocchio (Rocchio 1971). Rocchio suggested the following algorithm for relevance feedback, where $\vec{q}_{\text{old}}$ is the original query, $\vec{q}_{\text{new}}$ is the revised query, $\vec{d}^{(i)}_{\text{rel}}(1 \leq i \leq r)$ is one of the r documents the user selected as relevant, and $\vec{d}^{(i)}_{\text{nonrel}}(1 \leq i \leq n)$ is one of the n documents the user selected as non-relevant.

$$\vec{q}_{\text{new}} \;=\; \vec{q}_{\text{old}} + \frac{1}{r}\sum_{i=1}^{r}\vec{d}^{\,(i)}_{\text{rel}} - \frac{1}{n}\sum_{i=1}^{n}\vec{d}^{\,(i)}_{\text{nonrel}} \tag{1.4}$$

The normalised vectors of documents and queries can be viewed as points on a hypersphere at unit length from the origin. In Equation (1.4), 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 minimised. 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.

1.3.3 Term weighting and other caveats

The main disadvantage of the vector space model is that it does not in any way define what the values of the vector components should be. The problem of assigning appropriate values to the vector components is known as *term weighting*. Early experiments by Salton (1971) and Salton and Yang (1973) showed that term weighting is not a trivial problem at all. They suggested so-called *tf.idf* weights, a combination of term frequency *tf*, which is the number of occurrences of a term in a document, and *idf*, the inverse document frequency, which is a value inversely related to the document frequency *df*, which is the number of documents that contain the term. Many modern weighting algorithms are versions of the family of *tf.idf* weighting algorithms. Salton's original *tf.idf* weights perform relatively poorly, in some cases worse than simple *idf* weighting. They are defined as:

$$d_k = q_k = \mathit{tf}(k, d) \cdot \log \frac{N}{df(k)} \tag{1.5}$$

where $\mathit{tf}(k, d)$ is the number of occurrences of the term k in the document d, $\mathit{df}(k)$ is the number of documents containing k, and N is the total number of documents in the collection. Another problem with the vector space model is its implementation. The calculation of the cosine measure needs the values of all vector components, but these are not available in an inverted file. In practice, the normalised values and the vector product algorithm have to be used. Either the normalised weights have to be stored in the inverted file, or the normalisation values have to be stored separately. Both are problematic in case of incremental updates of the index. Adding a single new document changes the document frequencies of terms that occur in the document, which changes the vector lengths of every document that contains one or more of these terms.

1.4 Probabilistic Approaches

Several approaches that try to define term weighting more formally are based on *probability theory*. The notion of the probability of something, for instance the probability of relevance notated as $P(R)$, is usually formalised through the concept of an experiment, where an experiment is the process by which an observation is made. The set of all possible outcomes of the experiment is called the sample space. In the case of $P(R)$ the sample space might be {*relevant*, *irrelevant*}, and we might define the random variable R to take the values {0, 1}, where 0 = irrelevant and 1 = relevant.

Let's define an experiment for which we take one document from the collection at random. If we know the number of relevant documents in the collection, say 100 documents are relevant, and we know the total number of documents in the collection, say 1 million, then the quotient of those two defines the probability of relevance $P(R=1) = 100/1000\,000 = 0.0001$. Suppose furthermore that $P(D_k)$ is the probability that a document contains the term k with the sample space {0, 1}, (0 = the document does not contain term k, 1 = the document contains term k), then we will use $P(R, D_k)$ to denote the *joint probability distribution* with outcomes {(0, 0), (0, 1), (1, 0) and (1, 1)}, and we will use $P(R|D_k)$ to denote the *conditional probability distribution* with outcomes {0, 1}. So, $P(R=1|D_k=1)$ is the probability of relevance if we consider documents that contain the term k.

Note that the notation $P(\ldots)$ is overloaded. Whenever we are talking about a different random variable or sample space, we are also talking about a different measure P. So, one equation might refer to several probability measures, all ambiguously referred to as P. Also note that random variables such as D and T might have different sample spaces in different models. For instance, D in the probabilistic indexing model is a random variable denoting '*this* is the relevant document', that has as possible outcomes the identifiers of the documents in the collection. However, D in the probabilistic retrieval model is a random variable that has as possible outcomes all possible document descriptions, which in this case are vectors with binary components d_k that denote whether a document is indexed by term k or not.

1.4.1 The probabilistic indexing model

As early as 1960, Bill Maron and Larry Kuhns (Maron and Kuhns 1960) defined their probabilistic indexing model. Unlike Luhn, they did not target automatic indexing by information retrieval systems. Manual indexing was still guiding the field, so they suggested that a human indexer, who runs through the various index terms T that possibly apply to a document D, assigns a probability $P(T|D)$ to a term given a document instead of making a yes/no decision for each term. So, every document ends up with a set of possible index terms, weighted by $P(T|D)$, where $P(T|D)$ is the probability that, if a user wants information of the kind contained in document D, he/she will formulate a query by using T. Using Bayes' rule, i.e.

$$P(D|T) = \frac{P(T|D)P(D)}{P(T)} \tag{1.6}$$

they then suggest to rank the documents by $P(D|T)$, that is, the probability that the document D is relevant, given that the user formulated a query by using the term T. Note that $P(T)$ in the denominator of the right-hand side is constant for any given query term T, and consequently documents might be ranked by $P(T|D)P(D)$ which is a quantity proportional to the value of $P(D|T)$. In the formula, $P(D)$ is the *a priori* probability of relevance of document D.

Whereas $P(T|D)$ is defined by the human indexer, Maron and Kuhns suggest that $P(D)$ can be defined by statistics on document usage, i.e. by the quotient of the number of uses of document D by the total number of document uses. So, their usage of the document prior $P(D)$ can be seen as the very first description of popularity ranking, which became important for internet search (see Section 1.4.6). Interestingly, an estimate of $P(T|D)$ might be obtained in a similar way by storing, for each use of a document, also the query term that was entered to retrieve the document in the first place. Maron and Kuhns state that 'such a procedure would of course be extremely impractical', but

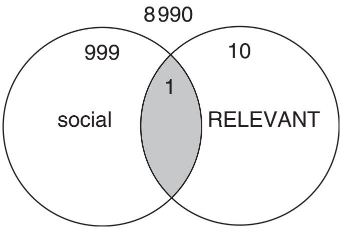

Figure 1.5 Venn diagram of the collection given the query term `social`

in fact, such techniques – rank optimization using so-called click-through rates – are now common in web search engines as well (Joachims *et al*. 2005). Probabilistic indexing models were also studied by Fuhr (1989).

1.4.2 The probabilistic retrieval model

Whereas Maron and Kuhns introduced ranking by the probability of relevance, it was Stephen Robertson who turned the idea into a principle. He formulated the probability ranking principle, which he attributed to William Cooper, as follows (Robertson 1977).

> If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.

This seems a rather trivial requirement indeed, since the objective of information retrieval systems is defined in Section 1.1 as to 'assist users in finding the information they need', but its implications might be very different from Luhn's similarity principle. Suppose a user enters a query containing a single term, for instance the term `social`. If all documents that fulfil the user's need were known, it would be possible to divide the collection into four non-overlapping document sets as visualised in the Venn diagram of Figure 1.5. The figure contains additional information about the size of each of the non-overlapping sets. Suppose the collection in question has 10 000 documents, of which 1000 contain the word 'social'. Furthermore, suppose that 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: $10/9000 = 0.0011$. Based on this evidence, the best performance is achieved if the system returns 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.

Stephen Robertson and Karen Spärck-Jones based their probabilistic retrieval model on this line of reasoning (Robertson and Spärck-Jones 1976). They suggested to rank documents by $P(R|D)$, that is the probability of relevance R given the document's content description D. Note that D is here a vector of binary components, each component typically representing a term, whereas in the previous section D was the 'relevant document'. In the probabilistic retrieval model the probability $P(R|D)$ has to be interpreted as follows: there might be several, say 10, documents that are represented by the same D. If 9 of them are relevant, then $P(R|D) = 0.9$. To make this work in practice, we use Bayes' rule on the probability odds $P(R|D)/P(\overline{R}|D)$, where $\overline{R}$ denotes irrelevance. The odds allow us to ignore $P(D)$ in the computation while still providing a ranking by the probability of relevance.

Additionally, we assume independence between terms given relevance.

$$\frac{P(R|D)}{P(\overline{R}|D)} = \frac{P(D|R)P(R)}{P(D|\overline{R})P(\overline{R})} = \frac{\prod_k P(D_k|R)P(R)}{\prod_k P(D_k|\overline{R})P(\overline{R})} \tag{1.7}$$

Here, D_k denotes the kth component (term) in the document vector. The probabilities of the terms are defined as above from examples of relevant documents, that is, in Figure 1.5, the probability of `social` given relevance is $1/11$. A more convenient implementation of probabilistic retrieval uses the following three order-preserving transformations. First, the documents are ranked by sums of logarithmic odds, instead of the odds themselves. Second, the *a priori* odds of relevance $P(R)/P(\overline{R})$ is ignored. Third, we subtract $\sum_k \log(P(D_k = 0|R)/P(D_k = 0|\overline{R}))$, i.e. the score of the empty document, from all document scores. This way, the sum over all terms, which might be millions of terms, only includes non-zero values for terms that are present in the document.

$$\text{matching-score}(D) = \sum_{k \in \text{ matching terms}} \log \frac{P(D_k = 1|R)\, P(D_k = 0|\overline{R})}{P(D_k = 1|\overline{R}) P(D_k = 0|R)} \tag{1.8}$$

In practice, terms that are not in the query are also ignored in Equation (1.8). Making full use of the probabilistic retrieval model requires two things: examples of relevant documents and long queries. Relevant documents are needed to compute $P(D_k|R)$, that is, the probability that the document contains the term k given relevance. Long queries are needed because the model only distinguishes term presence and term absence in documents and as a consequence, the number of distinct values of document scores is low for short queries. For a one-word query, the number of distinct probabilities is two (either a document contains the word or not), for a two-word query it is four (the document contains both terms, or only the first term, or only the second, or neither), for a three-word query it is eight, etc. Obviously, this makes the model inadequate for web search, for which no relevant documents are known beforehand and for which queries are typically short. However, the model is helpful in, for instance, spam filters. Spam filters accumulate many examples of relevant (no spam or 'ham') and irrelevant (spam) documents over time. To decide if an incoming email is spam or ham, the full text of the email can be used instead of a just few query terms.

1.4.3 The 2-Poisson model

Bookstein and Swanson (1974) studied the problem of developing a set of statistical rules for the purpose of identifying the index terms of a document. They suggested that the number of occurrences *tf* of terms in documents could be modelled by a mixture of two Poisson distributions as follows, where X is a random variable for the number of occurrences.

$$P(X = tf) = \lambda \frac{e^{-\mu_1}(\mu_1)^{tf}}{tf!} + (1 - \lambda) \frac{e^{-\mu_2}(\mu_2)^{tf}}{tf!} \tag{1.9}$$

The model assumes that the documents were created by a random stream of term occurrences. For each term, the collection can be divided into two subsets. Documents in subset one treat a subject referred to by a term to a greater extent than documents in subset two. This is represented by λ which is the proportion of the documents that belong to subset one and by the Poisson means μ_1 and $\mu_2 (\mu_1 \geq \mu_2)$ which can be estimated from the mean number of occurrences of the term in the respective subsets. For each term, the model needs these three parameters, but unfortunately, it is unknown to which subset each document belongs. The estimation of the three parameters should therefore be done iteratively by applying, e.g. the expectation maximisation algorithm (Dempster *et al*. 1977) or alternatively by the method of moments, as done by Harter (1975).

If a document is taken at random from subset one, then the probability of relevance of this document is assumed to be equal to, or higher than, the probability of relevance of a document from subset two;

because the probability of relevance is assumed to be correlated with the extent to which a subject referred to by a term is treated, and because $\mu_1 \geq \mu_2$. Useful terms will make a good distinction between relevant and non-relevant documents, that is, both subsets will have very different Poisson means μ_1 and μ_2. Therefore, Harter (1975) suggests the following measure of effectiveness of an index term that can be used to rank the documents given a query.

$$z = \frac{\mu_1 - \mu_2}{\sqrt{\mu_1 + \mu_2}} \tag{1.10}$$

The 2-Poisson model's main advantage is that it does not need an additional term weighting algorithm to be implemented. In this respect, the model contributed to the understanding of information retrieval and inspired some researchers in developing new models, as shown in the next paragraph. The model's biggest problem, however, is the estimation of the parameters. For each term there are three unknown parameters that cannot be estimated directly from the observed data. Furthermore, despite the model's complexity, it still might not fit the actual data if the term frequencies differ very much per document. Some studies therefore examine the use of more than two Poisson functions, but this makes the estimation problem even more intractable (Margulis 1993).

Robertson *et al.* (1981) proposed to use the 2-Poisson model to include the frequency of terms within documents in the probabilistic model. Although the actual implementation of this model is cumbersome, it inspired Stephen Robertson and Stephen Walker in developing the Okapi BM25 term weighting algorithm, which is still one of the best performing term weighting algorithms (Robertson and Walker 1994; Spärck-Jones *et al.* 2000).

1.4.4 Bayesian network models

In 1991, Howard Turtle proposed the *inference network model* (Turtle and Croft 1991) which is formal in the sense that it is based on the Bayesian network mechanism (Metzler and Croft 2004). A Bayesian network is an acyclic directed graph (a directed graph is acyclic if there is no directed path $A \to \cdots \to Z$ such that $A = Z$) that encodes probabilistic dependency relationships between random variables. The presentation of probability distributions as directed graphs, makes it possible to analyse complex conditional independence assumptions by following a graph theoretic approach. In practice, the inference network model is comprised of four layers of nodes: document nodes, representation nodes, query nodes and the information need node. Figure 1.6 shows a simplified inference network model. All nodes in the network represent binary random variables with values {0, 1}. To see how the model works in theory, it is instructive to look at a subset of the nodes, for instance the nodes r_2, q_1, q_3 and I, and ignore the other nodes for the moment. By the chain rule of probability, the joint probability of the nodes r_2, q_1, q_3 and I is:

$$P(r_2, q_1, q_3, I) = P(r_2)P(q_1|r_2)P(q_3|r_2, q_1)P(I|r_2, q_1, q_3) \tag{1.11}$$

The directions of the arcs suggest the dependence relations between the random variables. The event 'information need is fulfilled' ($I = 1$) has two possible causes: query node q_1 is true, or query node q_3 is true (remember we are ignoring q_2). The two query nodes in turn depend on the representation node r_2. So, the model makes the following conditional independence assumptions.

$$P(r_2, q_1, q_3, I) = P(r_2)\,P(q_1|r_2)\,P(q_3|r_2)\,P(I|q_1, q_3) \tag{1.12}$$

On the right-hand side, the third probability measure is simplified because q_1 and q_3 are independent, given their parent r_2. The last part $P(I|q_1, q_3)$ is simplified because I is independent of r_2, given its parents q_1 and q_3.

Straightforward use of the network is impractical if there are a large number of query nodes. The number of probabilities that have to be specified for a node grows exponentially with its number

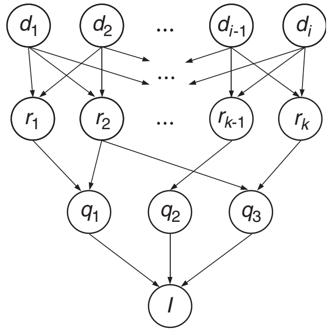

Figure 1.6 A simplified inference network

of parents. For example, a network with n query nodes requires the specification of 2^{n+1} possible values of $P(I|q_1, q_2, \cdots, q_n)$ for the information need node. For this reason, all network layers need some form of approximation. Metzler and Croft (2004) describe the following approximations. For the document layer they assume that only a single document is observed at a time, and for every single document a separate network is constructed for which the document layer is ignored. For the representation layer of every network, the probability of the representation nodes (which effectively are priors now, because the document layer is ignored) are estimated by some retrieval model. Note that a representation node is usually a single term, but it might also be a phrase. Finally, the query nodes and the information need node are approximated by standard probability distributions defined by so-called believe operators. These operators combine probability values from representation nodes and other query nodes in a fixed manner. If the values of $P(q_1|r_2)$, and $P(q_3|r_2)$ are given by p_1 and p_2, then the calculation of $P(I|r_2)$ might be done by operators such as 'and', 'or', 'sum', and 'wsum'.

$$\begin{aligned} P_{\text{and}}(I|r_2) &= p_1 \cdot p_2 \\ P_{\text{or}}(I|r_2) &= 1 - ((1 - p_1)(1 - p_2)) \\ P_{\text{sum}}(I|r_2) &= (p_1 + p_2)/2 \\ P_{\text{wsum}}(I|r_2) &= w_1\, p_1 + w_2\, p_2 \end{aligned} \tag{1.13}$$

It can be shown that, for these operators, so-called link matrices exists, that is, for each operator there exists a definition of for instance $P(I|q_1, q_3)$ that can be computed, as shown in Equation (1.13). So, although the link matrix that belongs to the operator may be huge, it does not exist in practice and its result can be computed in linear time. One might argue though, that the approximations on each network layer make it questionable if the approach still deserves to be called a 'Bayesian network model'.

1.4.5 Language models

Language models were applied to information retrieval by a number of researchers in the late 1990s (Ponte and Croft 1998; Hiemstra and Kraaij 1998; Miller *et al.* 1999). They originate from probabilistic models of language generation developed for automatic speech recognition systems in the early 1980s (e.g. Rabiner 1990). Automatic speech recognition systems combine probabilities of two distinct models: the acoustic model, and the language model. The acoustic model might for instance produce the following candidate texts in decreasing order of probability: 'food born thing', 'good corn sing', 'mood morning', and 'good morning'. Now, the language model would determine that the phrase 'good morning' is much more probable, i.e. it occurs more frequently in English than the other phrases. When combined with the acoustic model, the system is able to decide that 'good morning' was the most likely utterance, thereby increasing the system's performance.

For information retrieval, language models are built for each document. By following this approach, the language model of the book you are reading now would assign an exceptionally high probability to the word 'retrieval', indicating that this book would be a good candidate for retrieval if the query contains this word. Language models take the same starting point as the probabilistic indexing model by Maron and Kuhns described in Section 1.4.1. That is, given D – the document is relevant – the user will formulate a query by using a term T with some probability $P(T|D)$. The probability is defined by the text of the documents. If a certain document consists of 100 words, and of those the word 'good' occurs twice, then the probability of 'good', given that the document is relevant, is simply defined as 0.02. For queries with multiple words, we assume that query words are generated independently from each other, i.e. the conditional probabilities of the terms $T_1, T_2, \cdots$ given the document are multiplied:

$$P(T_1, T_2, \cdots |D) = \prod_i P(T_i|D) \tag{1.14}$$

As a motivation for using the probability of the query given the document, one might think of the following experiment. Suppose we ask one million monkeys to pick a good three-word query for several documents. Each monkey will point three times at random to each document. Whatever word the monkey points to, will be the (next) word in the query. Suppose that seven monkeys accidentally pointed to the words 'information', 'retrieval' and 'model' for document 1, and only two monkeys accidentally pointed to these words for document 2. Then, document 1 would be a better document for the query 'information retrieval model' than document 2.

The above experiment assigns zero probability to words that do not occur anywhere in the document, and because we multiply the probabilities of the single words, it assigns zero probability to documents that do not contain all of the words. For some applications this is not a problem. For instance for a web search engine, queries are usually short and it will rarely happen that no web page contains all query terms. For many other applications empty results happen much more often, which might be problematic for the user. Therefore, a technique called *smoothing* is applied. Smoothing assigns some non-zero probability to unseen events. One approach to smoothing takes a linear combination of $P(T_i|D)$ and a background model $P(T_i)$ as follows.

$$P(T_1, \cdots, T_n|D) = \prod_{i=1}^{n} (\lambda P(T_i|D) + (1-\lambda)P(T_i)) \tag{1.15}$$

The background model $P(T_i)$ might be defined by the probability of term occurrence in the collection, i.e. by the quotient of the total number of occurrences in the collection divided by the length of the collection. In the equation, λ is an unknown parameter that has to be set empirically. Linear interpolation smoothing accounts for the fact that some query words do not seem to be related to the relevance of documents at all. For instance in the query 'capital of the Netherlands', the words 'of' and 'the' might be seen as words from the user's general English vocabulary, and not as words from the relevant document he/she is looking for. In terms of the experiment above, a monkey would either pick a word at random from the document with probability λ or the monkey would pick a word at random from the entire collection. A more convenient implementation of the linear interpolation models can be achieved with order-preserving transformations that are similar to those for the probabilistic retrieval model (see Equation 1.8). We multiply both sides of the equation by $\prod_i (1-\lambda)P(T_i)$ and take the logarithm, which leads to:

$$\text{matching score}(d) = \sum_{k \in \text{ matching terms}} \log\left(1 + \frac{tf(k,d)}{\sum_t tf(t,d)} \cdot \frac{\sum_t cf(t)}{cf(k)} \cdot \frac{\lambda}{1-\lambda}\right) \tag{1.16}$$

where, $P(T_i = t_i) = cf(t_i)/\sum_t cf(t)$, and $cf(t) = \sum_d tf(t,d)$.

There are many approaches to smoothing, most pioneered for automatic speech recognition (Chen and Goodman 1996). Another approach to smoothing that is often used for information retrieval is so-called Dirichlet smoothing, which is defined as (Zhai and Lafferty 2004):

$$P(T_1 = t_1, \cdots, T_n = t_n | D = d) = \prod_{i=1}^{n} \frac{tf(t_i, d) + \mu P(T_i = t_i)}{(\sum_t tf(t, d)) + \mu} \tag{1.17}$$

Here, μ is a real number $\mu \geq 0$. Dirichlet smoothing accounts for the fact that documents are too small to reliably estimate a language model. Smoothing by Equation (1.17) has a relatively big effect on small documents, but relatively small effect on bigger documents.

The equations above define the probability of a query given a document, but obviously, the system should rank by the probability of the documents given the query. These two probabilities are related by Bayes' rule as follows.

$$P(D|T_1, T_2, \cdots, T_n) = \frac{P(T_1, T_2, \cdots, T_n|D) P(D)}{P(T_1, T_2, \cdots, T_n)} \tag{1.18}$$

The left-hand side of Equation (1.18) cannot be used directly because the independence assumption presented above assumes terms are independent, given the document. So, in order to compute the probability of the document D given the query, we need to multiply Equation (1.15) by $P(D)$ and divide it by $P(T_1, \cdots, T_n)$. Again, as stated in the previous paragraph, the probabilities themselves are of no interest, only the ranking of the document by the probabilities is. And since $P(T_1, \cdots, T_n)$ does not depend on the document, ranking the documents by the numerator of the right-hand side of Equation (1.18) will rank them by the probability given the query. This shows the importance of $P(D)$, the marginal probability, or prior probability of the document, i.e. it is the probability that the document is relevant if we do not know the query (yet). For instance, we might assume that long documents are more likely to be useful than short documents. In web search, such so-called static rankings (see Section 1.4.6), are commonly used. For instance, documents with many links pointing to them are more likely to be relevant, as shown in the next section.

1.4.6 Google's PageRank model

When Sergey Brin and Lawrence Page launched the web search engine Google in 1998 (Brin and Page 1998), it had two features that distinguished it from other web search engines. It had a simple no-nonsense search interface, and, it used a radically different approach to rank the search results. Instead of returning documents that closely match the query terms (i.e. by using any of the models in the preceding sections), they aimed at returning high-quality documents, i.e. documents from trusted sites. Google uses the hyperlink structure of the web to determine the quality of a page, called *PageRank*. Web pages that are linked at from many places around the web are probably worth looking at: they must be *high-quality* pages. If pages that have links from other high-quality web pages, for instance DMOZ or Wikipedia[1], then that is a further indication that they are likely to be worth looking at. The PageRank of a page d is defined as $P(D = d)$, i.e. the probability that d is relevant as used in the probabilistic indexing model in Section 1.4.1 and as used in the language modelling approach in Section 1.4.5. It is defined as:

$$P(D{=}d) = (1{-}\lambda)\frac{1}{\#\text{pages}} + \lambda \sum_{i|i \text{ links to } d} P(D{=}i) P(D{=}d|D{=}i) \tag{1.19}$$

If we ignore $(1{-}\lambda)/\#\text{pages}$ for the moment, then the PageRank $P(D{=}d)$ is recursively defined as the sum of the PageRanks $P(D{=}i)$ of all pages i that link to d, multiplied by the probability $P(D{=}d|D{=}i)$ of following a link from i to d. One might think of the PageRank as the probability that a random surfer visits a page. Suppose we ask the monkeys from the previous section to surf the

[1] See http://dmoz.org and http://wikipedia.org

web from a randomly chosen starting point i. Each monkey will now click on a random hyperlink with the probability $P(D = d|D = i)$ which is defined as one divided by the number of links on page i. This monkey will end up in d. But other monkeys might end up in d as well: those that started on another page that happens to link to d. After letting the monkeys surf a while, the highest-quality pages, i.e. the best-connected pages, will have most monkeys that look at it.

The above experiment has a similar problem with zero probabilities as the language modelling approach. Some pages might have no links pointing to them, so they will get a zero PageRank. Others might not link to any other page, so you cannot leave the page by following hyperlinks. The solution is also similar to the zero probability problem in the language modelling approach: we smooth the model by some background model, in this case the background is uniformly distributed over all pages. With some unknown probability λ a link is followed, but with probability $1 - \lambda$ a random page is selected, which is like a monkey typing in a random (but valid) URL.

PageRank is a so-called static ranking function, that is, it does not depend on the query. It is computed once off-line at indexing time by iteratively calculating the page rank of pages at time $t + 1$ from the page ranks as calculated in a previous interations at time t until they do not change significantly anymore. Once the PageRank of every page is calculated it can be used during querying. One possible way to use PageRank during querying is as follows: select the documents that contain all query terms (i.e. a Boolean AND query) and rank those documents by their page rank. Interestingly, a simple algorithm such as this would not only be effective for web search, it can also be implemented very efficiently (Richardson *et al*. 2006). In practice, web search engines such as Google use many more factors in their ranking than just page rank alone. In terms of the probabilistic indexing model and the language modelling approaches, static rankings are simply document priors, i.e. the *a priori* probability of the document being relevant, that should be combined with the probability of terms given the document. Document priors can be easily combined with standard language modelling probabilities and are as such powerful means to improve the effectiveness of, for instance, queries for home pages in web search (Kraaij *et al*. 2002).

1.5 Summary and Further Reading

There is no such thing as a dominating model or theory of information retrieval, unlike the situation in, for instance, the area of databases where the relational model is *the* dominating database model. In information retrieval, some models work for some applications, whereas others work for other applications. For instance, the region models introduced in Section 1.2.2 have been designed to search in semi-structured data; the vector space models in Section 1.3 are well suited for similarity search and relevance feedback in many (also non-textual) situations if a good weighting function is available; the probabilistic retrieval model of Section 1.4.2 might be a good choice if examples of relevant and non-relevant documents are available; language models in Section 1.4.5 are helpful in situations that require models of language similarity or document priors; and the PageRank model of Section 1.4.6 is often used in situations that need modelling of more of less static relations between documents. This chapter describes these and other information retrieval models in a tutorial style in order to explain the consequences of modelling assumptions. Once the reader is aware of the consequences of modelling assumptions, he or she will be able to choose a model of information retrieval that is adequate in new situations.

Whereas the citations in the text are helpful for background information and for putting things in a historical context, we recommend the following publications for people interested in a more in-depth treatment of information retrieval models. A good starting point for the region models of Section 1.2.2 would be the overview article of Hiemstra and Baeza-Yates (2009). An in-depth description of the vector space approaches of Section 1.3, including, for instance, latent semantic indexing is given by Berry *et al*. (1999). The probabilistic retrieval model of Section 1.4.2 and developments based on the model are well described by Spärck-Jones *et al*. (2000); de Campos *et al*. (2004) edited a special issue on Bayesian networks for information retrieval as described in Section 1.4.4. An excellent overview of statistical language models for information retrieval is given by Zhai (2008). Finally, a good follow-up article on Google's PageRank is given by Henzinger (2001).

Exercises

1.1 In the Boolean model of Section 1.2.1, there are a large number of queries that can be formulated with three query terms, for instance `one OR two OR three`, or `(one OR two) AND three`, or possibly `(one AND three) OR (two AND three)`. Some of these queries, however, for instance the last two queries, return the same set of documents. *How many different sets of documents can be specified given three query terms?* Explain your answer.

(a) 8
(b) 9
(c) 256
(d) unlimited

1.2 Given a query $\vec{q}$ and a document $\vec{d}$ in the vector space model of Section 1.3.1. Suppose the similarity between $\vec{q}$ and $\vec{d}$ is 0.08. Suppose we interchange the full contents of the document with the query, that is, all words from $\vec{q}$ go to $\vec{d}$ and all words from $\vec{d}$ go to $\vec{q}$. *What will now be the similarity between $\vec{q}$ and $\vec{d}$?* Explain your answer.

(a) smaller than 0.08
(b) equal: 0.08
(c) bigger than 0.08
(d) it depends on the term weighting algorithm

1.3 In the vector approach of Section 1.3 that uses Equation (1.1) and *tf.idf* term weighting, suppose we add some documents to the collection. *Do the weights of the terms in the documents that were indexed before change?* Explain your answer.

(a) no
(b) yes, it affects the *tf*'s of terms in other documents
(c) yes, it affects the *idf*'s of terms in other documents
(d) yes, it affects the *tf*'s and the *idf*'s of terms in other documents

1.4 In the vector space model using the cosine similarity of Equation (1.2) and *tf.idf* term weighting, suppose we add some documents to the collection. *Do the weights of the terms in the documents that were indexed before change?* Explain your answer.

(a) no, other documents are unaffected
(b) yes, the same weights as in Exercise 1.3
(c) yes, more weights change than in Exercise 1.3, but not all
(d) yes, all weights in the index change

1.5 In the probabilistic model of Section 1.4.2, two documents might get the same score. *How many different scores do we expect to get if we enter three query terms?* Explain your answer.

(a) 8
(b) 9
(c) 256
(d) unlimited

1.6 For the probabilistic retrieval model of Section 1.4.2, suppose we query for the word *retrieval*, and document D has more occurrences of *retrieval* than document E. *Which document will be ranked first?* Explain your answer.

(a) D will be ranked before E
(b) E will be ranked before D

(c) it depends on the model's implementation
(d) it depends on the lengths of D and E

1.7 In the language modeling approach of Section 1.4.5, suppose the model does not use smoothing. In the case we query for the word *retrieval*, and document D consisting of 100 words in total, contains the word *retrieval* 4 times. *What is* $P(T = retrieval|D)$*?*

(a) smaller than $4/100 = 0.04$
(b) equal to $4/100 = 0.04$
(c) bigger than $4/100 = 0.04$
(d) it depends on the term weighting algorithm

1.8 In the language modeling approach of Section 1.4.5, suppose we use a linear combination of a document model and a collection model as in Equation (1.15). *What happens if we take* $\lambda = 1$*?* Explain your answer.

(a) all docucments get a probability > 0
(b) documents that contain at least one query term get a probability > 0
(c) only documents that contain all query terms get a probability > 0
(d) the system returns a randomly ranked list

References

Berry, M. W, Z. Drmac and E. R. Jessup (1999). Matrices, vector spaces, and information retrieval. *SIAM Review* **41**(2) 335–362.

Bookstein, A. and D. Swanson (1974). Probabilistic models for automatic indexing. *Journal of the American Society for Information Science* **25**(5) 313–318.

Brin, S. and L. Page (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* **30**(1–7) 107–117.

Burkowski, F. (1992). Retrieval activities in a database consisting of heterogeneous collections of structured texts. In *Proceedings of the 15th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)* pp. 112–125.

Campos, L. M. de, J. M. Fernandez-Luna and J. F. Huete (eds) (2004). Special issue on Bayesian networks and information retrieval. *Information Processing and Management* **40**(5).

Chen, S. and J. Goodman (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 310–318.

Clarke, C., G. Cormack, and F. Burkowski (1995). Algebra for structured text search and a framework for its implementation. *The Computer Journal* **38**(1) 43–56.

Dempster, A., N. Laird, and D. Rubin (1977). Maximum likelihood from incomplete data via the em-algorithm plus discussions on the paper. *Journal of the Royal Statistical Society* **39**(B) 1–38.

Fuhr, N. (1989). Models for retrieval with probabilistic indexing. *Information processing and management* **25**(1) 55–72.

Harter, S. (1975). An algorithm for probabilistic indexing. *Journal of the American Society for Information Science* **26**(4) 280–289.

Henzinger, M. R. (2001) Hyperlink Analysis for the Web. *IEEE Internet Computing* **5**(1) 45–50.

Hiemstra, D. and R. Baeza-Yates (2009). Structured text retrieval models. In M. Tamer zsu and Ling Liu (eds), *Encyclopedia of Database Systems*, Springer.

Hiemstra, D. and W. Kraaij (1998). Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the 7th Text Retrieval Conference TREC-7*, pp. 227–238. NIST Special Publication 500-242.

Jaakkola, J. and P. Kilpelainen (1999). Nested text-region algebra. *Technical Report CR-1999-2*, Department of Computer Science, University of Helsinki.

Joachims, T., L. Granka, B. Pan, H. Hembrooke and G. Gay (2005). Accurately interpreting click-through data as implicit feedback. In *Proceedings of the 28th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*, pp. 154–161.

Kraaij, W., T. Westerveld, and D. Hiemstra (2002). The importance of prior probabilities for entry page search. In *Proceedings of the 25th ACM Conference on Research and Development in Information Retrieval (SIGIR'02)*, pp. 27–34.

Luhn, H. (1957). A statistical approach to mechanised encoding and searching of litary information. *IBM Journal of Research and Development* **1**(4) 309–317.

Margulis, E. (1993). Modelling documents with multiple Poisson distributions. *Information Processing and Management* **29** 215–227.

Maron, M. and J. Kuhns (1960). On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery* **7** 216–244.

Metzler, D. and W. Croft (2004). Combining the language model and inference network approaches to retrieval. *Information Processing and Management* **40**(5) 735–750.

Mihajlovic, V. (2006). *Score Region Algebra: A flexible framework for structured information retrieval*. PhD Thesis, University of Twente.

Miller, D., T. Leek and R. Schwartz (1999). A hidden Markov model information retrieval system. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 214–221.

Navarro, G., and R. Baeza-Yates (1997). Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems* **15** 400–435.

Ponte, J. and W. Croft (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st ACM Conference on Research and Development in Information Retrieval (SIGIR'98)* pp. 275–281.

Rabiner, L. (1990). A tutorial on hidden markov models and selected applications in speech recognition. In A. Waibel and K. Lee (eds), *Readings in speech recognition*, pp. 267–296. Morgan Kaufmann.

Richardson, M., A. Prakash, and E. Brill (2006). Beyond PageRank: machine learning for static ranking. In *Proceedings of the 15th International Conference on World Wide Web*, pp. 707715–. ACM Press.

Robertson, S. (1977). The probability ranking principle in ir. *Journal of Documentation* **33**(4) 294–304.

Robertson, S. and K. Spärck-Jones (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science* **27** 129–146.

Robertson, S. and S. Walker (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR'94)*, pp. 232–241.

Robertson, S. E., C. J. van Rijsbergen, and M. F. Porter (1981). Probabilistic models of indexing and searching. In R. Oddy *et al.* (eds), *Information Retrieval Research*, pp. 35–56. Butterworths.

Rocchio, J. (1971). Relevance feedback in information retrieval. In G. Salton (ed.), *The Smart Retrieval System: Experiments in Automatic Document Processing*, pp. 313–323. Prentice Hall.

Salton, G. (1971). The SMART retrieval system: *Experiments in automatic document processing*. Prentice-Hall.

Salton, G. and E. A. Fox and H. Wu (1983). *Extended Boolean information retrieval*. *Communications of the ACM* **26**(11) 1022–1036.

Salton, G. and M. McGill (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Salton, G. and C. Yang (1973). On the specification of term values in automatic indexing. *Jounral of Documentation* **29**(4) 351–372.

Spärck-Jones, K., S. Walker, and S. Robertson (2000). A probabilistic model of information retrieval: Development and comparative experiments (part 1 and 2). *Information Processing and Management* **36**(6) 779–840.

Turtle, H. and W. Croft (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems* **9**(3) 187–222.

Zhai, C. and J. Lafferty (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems* **22**(2) 179–214.

Zhai, C. (2008). Statistical Language Models for Information Retrieval. *Foundations and Trends in Information Retrieval* **2**(3) 137–213.