11 Probabilistic information retrieval

We noted in Section 9.1.2 that if we have some known relevant and non-relevant documents, then we can straightforwardly start to estimate the probability of a term appearing in a relevant document $P(t_j|R=1)$, and that this could be the basis of a classifier that determines whether documents are relevant or not. In this chapter, we more systematically introduce this probabilistic approach to IR, which provides a different retrieval model, with different techniques for setting term weights, and a different way of thinking about modelling information retrieval.

There is more than one possible retrieval model which has a probabilistic basis. Here, we will introduce probability theory for IR and the Probability Ranking Principle (Sections 11.1–11.2), and then concentrate on the Binary Independence Model (Section 11.3), which is the original and still most influential probabilistic retrieval model. Finally, we will briefly touch on related but extended methods using term counts, including the empirically successful Okapi BM25 weighting scheme, and Bayesian Network models for IR (Section 11.4). In the next chapter, we then present the alternate probabilistic language modeling approach to IR, which has been developed with considerable success in recent years.

11.1 Probability in Information Retrieval

In this section, we motivate the use of probabilistic methods in IR, and review a few probability basics. In information retrieval, the user starts with an information need, which they translate into a query representation. Similarly, there are documents, each of which has been converted into a document representation (the latter differing at least by how text is tokenized and so on, but perhaps containing fundamentally less information, as when a non-positional index is used). Based on the two representations, we wish to determine how well a document satisfies an information need. In the Boolean or vector space models of IR, matching is done in a formally defined but semantically im-
precise calculus of index terms. Given only a query, an IR system has an uncertain understanding of the original information need. And given the query and document representations, a system also has an uncertain guess of whether a document has content relevant to a query. The whole idea of probability theory is to provide a principled foundation for reasoning under uncertainty. This chapter provides one answer as to how to exploit this foundation to estimate how likely it is that a document is relevant to a query.

We hope that the reader has seen a little basic probability theory previously, but to very quickly review, for events \( a \) and \( b \), there is the concept of a joint event \( P(a, b) \), where both are true, and a conditional probability, such as \( P(a|b) \), the probability of event \( a \) given that event \( b \) is true. Then the fundamental relationship between joint and conditional probabilities is given by:

\[
P(a, b) = P(a \cap b) = P(a|b)P(b) = P(b|a)P(a)
\]

Similarly,

\[
P(a, b) = P(b|a)P(a)
\]

\[
P(\neg a, b) = P(b|\neg a)P(\neg a)
\]

**Bayes’ Rule** From these we can derive Bayes’ Rule for inverting conditional probabilities:

\[
P(a|b) = \frac{P(b|a)P(a)}{\sum_{x \in \{a, \neg a\}} P(b|x)P(x)}P(a)
\]

This equation can also be thought of as a way of updating probabilities: we start with a prior probability \( P(a) \) and derive a posterior probability \( P(a|b) \) after having seen the evidence \( b \), based on the likelihood of \( b \) occurring. Finally, it is often useful to talk about the odds of an event, which provide a kind of multiplier for how probabilities change:

\[
\text{Odds: } O(a) = \frac{P(a)}{P(\neg a)} = \frac{P(a)}{1 - P(a)}
\]

**11.2 The Probability Ranking Principle**

Let \( d \) be a document in the collection. Let \( R \) represent that the document is relevant with respect to a given query \( q \), and let \( NR \) represent non-relevance. (This is the traditional notation. Using current notation for probabilistic models, it would be more standard to have a binary random variable \( R_{d,q} \) for relevance to a particular query \( q \), and then \( R \) would represent \( R_{d,q} = 1 \) and \( NR \) would represent \( R_{d,q} = 0 \).)

We assume the usual ranked retrieval setup, where there is a collection of documents, the user issues a query, and an ordered list of documents needs
to be returned. In this setup, the ranking method is the core of the IR system: in what order do we present the documents to the user. In a probabilistic model, the obvious answer to this question is to rank documents by the estimated probability of their relevance with respect to the information need. That is, we order documents \( d \) by \( P(R|d, q) \).

This is the basis of the Probability Ranking Principle (PRP) (van Rijsbergen 1979, 113–114):

“If a reference retrieval system’s response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.”

In the simplest case of the PRP, there are no selection costs or other utility concerns that would differentially weight actions or errors. The PRP in action then says to simply rank all documents by \( P(R|x, q) \). In this case the Bayes Optimal Decision Rule is to use a simple even odds threshold:

\[
x \text{ is relevant iff } P(R|x) > P(NR|x)
\]

**Theorem 11.1** Using the PRP is optimal, in the sense that it minimizes the loss (Bayes risk) under 1/0 loss.

Proof. Ripley (1996). The proof assumes that all probabilities are known correctly, which is never the case in practice.

In more complex models involving the PRP, we assume a model of retrieval costs. Let \( C_r \) be the cost of retrieval of a relevant document and \( C_{nr} \) the cost of retrieval of a non-relevant document. Then the Probability Ranking Principle says that if for a specific document \( x \) and for all documents \( x' \) not yet retrieved

\[
C_r \cdot P(R|x) + C_{nr} \cdot P(NR|x) \leq C_r \cdot P(R|x') + C_{nr} \cdot P(NR|x')
\]

then \( x \) is the next document to be retrieved. For the rest of this chapter, we will stick to the simple 1/0 loss case, and not further consider loss/utility models, but simply note that these give a formal framework where we can model differential costs of false positives and false negatives at the modeling stage, rather than simply at the evaluation stage, as discussed in Chapter 8.

To make a probabilistic retrieval strategy precise, we need to estimate how terms in documents contribute to relevance. Becoming more specific, we wish to know how statistics we can compute like term frequency, document frequency, and document length influence judgements about document relevance, and how they can be reasonably combined to estimate the probability
of document relevance. We then order documents by decreasing estimated probability of relevance.

What we wish to find is the probability $P(R|x)$ that a document $x$ is relevant. Using Bayes Rule, we have that:

$$P(R|x) = \frac{P(x|R)P(R)}{P(x)}$$  

$$P(NR|x) = \frac{P(x|NR)P(NR)}{P(x)}$$

Here, $P(x|R)$ and $P(x|NR)$ are the probability that if a relevant (or non-relevant, respectively) document is retrieved, then it is $x$. And $P(R)$ and $P(NR)$ indicate the prior probability of retrieving a relevant or non-relevant document respectively. If we knew the percentage of relevant documents in the collection, then we could estimate these quantities. Since a document must be either relevant or non-relevant to a query, we have that:

$$P(R|x) + P(NR|x) = 1$$

How do we compute all these probabilities? We never know the exact probabilities, and so we have to use estimates. The Binary Independence Model, which we present in the next section, makes some simple assumptions so as to provide easy ways of calculating the needed estimates. But before presenting this model in detail, let us note a few of the questionable assumptions of the model so far:

• Document relevance is again assumed to be independent. The relevance of each document is independent of the relevance of other documents. As we have noted, this is incorrect: it is especially harmful in practice if it allows a system to return duplicate or near duplicate documents.

• We are again working with just a Boolean model of relevance.

• The model assumes that the user has a single step information need. As discussed in Chapter 9, seeing a range of results might let the user refine their information need. Fortunately, as mentioned there, it is straightforward to extend the Binary Independence Model so as to provide a framework for relevance feedback, and we briefly present this model below.

11.3 The Binary Independence Model

The Binary Independence Model (BIM) which we present in this section is the model which has traditionally been used with the PRP. Here, “binary” means Boolean: documents are represented as Boolean term incidence vectors, just
as in the start of Chapter 1. A document is a vector \( \vec{x} = (x_1, \ldots, x_n) \) where \( x_i = 1 \) iff term \( i \) is present in document \( x \). Note that with this representation, many possible documents have the same vector representation. “Independence” means that terms are modeled as occurring in documents independently. The model recognizes no association between terms. This is far from correct, but a workable assumption: it is the “naive” assumption of Naive Bayes models; cf. Chapter 13. Indeed, the Binary Independence Model is just another name for the Bernoulli Naive Bayes model presented there.

### 11.3.1 Deriving a ranking function for query terms

Given a query \( q \), for each document \( d \), we need to compute \( P(R|q,d) \). In the BIM, the documents and queries are both represented as binary term incidence vectors. Let \( x \) be the binary term incidence vector representing \( d \). Then we wish to compute \( P(R|q,x) \). In theory, we should also replace the query with a binary representation, but in general the query inherently is in this representation, so we ignore this distinction. We are interested only in the ranking of documents. It is therefore equivalent to rank documents by their odds of relevance. Given Bayes’ Rule, we have:

\[
O(R|q, x) = \frac{P(R|q, x)}{P(NR|q, x)} = \frac{P(R|q)P(x|R, q)}{P(NR|q)P(x|NR, q)}
\]

So,

\[
O(R|q, x) = \frac{P(R|q)}{P(NR|q)} \cdot \frac{P(x|R, q)}{P(x|NR, q)}
\]

The left term on the right-hand side of Equation (11.11) is a constant for a given query. Since we are only ranking documents, there is thus no need for us to estimate it. The right-hand term does, however, require estimation. Using the independence (Naive Bayes) assumption, we have:

\[
\frac{P(x|R, q)}{P(x|NR, q)} = \prod_{i=1}^{N} \frac{P(x_i|R, q)}{P(x_i|NR, q)}
\]

So:

\[
O(R|q, d) = O(R|q) \cdot \prod_{i=1}^{N} \frac{P(x_i|R, q)}{P(x_i|NR, q)}
\]

Since each \( x_i \) is either 0 or 1, we can separate the terms to give:

\[
O(R|q, d) = O(R|q) \cdot \prod_{i:x_i=1} P(x_i = 1|R, q) \cdot P(x_i = 0|NR, q)
\]
Henceforth, let \( p_i = P(x_i = 1|R, q) \) and \( u_i = P(x_i = 1|NR, q) \).

Let us begin by assuming that for all terms not occurring in the query (i.e., \( q_i = 0 \)) that \( p_i = u_i \). That is, they are equally likely to occur in relevant and irrelevant documents. This assumption can be changed, for example when doing relevance feedback. Under this assumption, we need only consider terms in the products that appear in the query, and so,

\[
O(R|q, d) = O(R|q) \cdot \prod_{i:x_i=1} \frac{p_i}{u_i} \cdot \prod_{i:x_i=0} \frac{1 - p_i}{1 - u_i}
\]

The left product is over query terms found in the document and the right term is over query terms not found in the document.

We can manipulate this expression by including the query terms found in the document into the right product, but simultaneously dividing through by them in the left product, so the value is unchanged. Then we have:

\[
O(R|q, d) = O(R|q) \cdot \prod_{i:x_i=q_i=1} \frac{p_i(1 - u_i)}{u_i(1 - p_i)} \cdot \prod_{i:q_i=0} \frac{1 - p_i}{1 - u_i}
\]

The left product is still over query terms found in the document, but the right product is now over all query terms. That means that this right product is a constant for a particular query, just like the odds \( O(R|q) \). So the only quantity that needs to be estimated to rank documents for relevance to a query is the left product. We can equally rank documents by the log of this term, since \( \log \) is a monotonic function. The resulting quantity used for ranking is called the Retrieval Status Value (RSV) in this model:

\[
RSV = \log \prod_{i:x_i=q_i=1} \frac{p_i(1 - u_i)}{u_i(1 - p_i)} = \sum_{i:x_i=q_i=1} \log \frac{p_i(1 - u_i)}{u_i(1 - p_i)}
\]

So everything comes down to computing the RSV. Let \( c_i = \log \frac{p_i(1 - u_i)}{u_i(1 - p_i)} = \log p_i(1 - u_i) - \log u_i(1 - p_i) \). These \( c_i \) terms are log odds ratios for the terms in the query. How do we estimate these \( c_i \) quantities for a particular collection and query?

### 11.3.2 Probability estimates in theory

For each term \( i \), what would these numbers look like for the whole collection? Consider the following contingency table of counts of documents in the collection:

\[
\begin{array}{c|cc|c}
\text{Documents} & \text{Relevant} & \text{Non-relevant} & \text{Total} \\
\hline
x_i = 1 & s & n - s & n \\
x_i = 0 & S - s & (N - n) - (S - s) & N - n \\
\hline
\text{Total} & S & N - s & N \\
\end{array}
\]
Then the quantities we introduced earlier are: \( p_i = \frac{s}{S} \) and \( u_i = \frac{(n - s)}{(N - S)} \) and

\[
(11.19) \quad c_i = K(N, n, S, s) = \log \frac{s/(S - s)}{(n - s)/(N - n - (S - s))}
\]

To avoid the possibility of zeroes (such as if every or no relevant document has a term) it is fairly standard to add 0.5 to each of the quantities in the center 4 terms, and then to adjust the marginal counts (the totals) accordingly (so, the bottom right cell totals \( N + 2 \)). We will discuss further methods of smoothing estimated counts in Chapter 12, but this method will do for now. Then we have that:

\[
(11.20) \quad \hat{c}_i = K(N, n, S, s) = \log \frac{(s + 0.5)/(S - s + 0.5)}{(n - s + 0.5)/(N - n - S + s + 0.5)}
\]

### 11.3.3 Probability estimates in practice

Under the assumption that relevant documents are a very small percentage of the collection, it is plausible to approximate statistics for non-relevant documents by statistics from the whole collection. Under this assumption, \( u_i \) (the probability of term occurrence in non-relevant documents for a query) is \( n/N \) and

\[
(11.21) \quad \log[(1 - u_i)/u_i] = \log[(N - n)/n] \approx \log N/n
\]

In other words, we can provide a theoretical justification for the most frequently used form of idf weighting.

The above approximation technique cannot easily be extended to relevant documents. The quantity \( p_i \) (the probability of term occurrence in relevant documents) can be estimated in various ways:

- From the frequency of term occurrence in known relevant documents (if we know some). This is the basis of probabilistic approaches to relevance weighting in a relevance feedback loop, discussed below.
- As a constant, as was proposed in the Croft and Harper (1979) combination match model. For instance, we might assume that \( p_i \) is constant over all terms \( x_i \) in the query and that \( p_i = 0.5 \). This means that each term has even odds of appearing in a relevant document, and so the \( p_i \) and \((1 - p_i)\) factors cancel out in the expression for RSV. Such an estimate is weak, but doesn’t violently disagree with our hopes for the search terms appearing in documents. Combining this method with our earlier approximation for \( u_i \), the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting. For short documents
It can be made proportional to the log of the probability of occurrence of the term in the collection (Greiff 1998).

- Iterative methods of estimation, which combine some of the above ideas, are discussed below.

11.3.4 Probabilistic approaches to relevance feedback

Accurate estimation of \( p_i \) points in the direction of using (pseudo-)relevance feedback, perhaps in an iterative process of estimation. The probabilistic approach to relevance feedback works as follows:

1. Guess a preliminary probabilistic description of \( R \) and use it to retrieve a first set of documents \( V \). This can use the probability estimates of the previous section.

2. We interact with the user to refine the description. We learn from the user some definite subset members of \( VR \subset R \) and \( VNR \subset NR \).

3. We reestimate \( p_i \) and \( u_i \) on the basis of known relevant and irrelevant documents. If the sets \( VR \) and \( VNR \) are large enough, we might estimate these quantities directly from these documents as maximum likelihood estimates:

\[
p_i = \frac{|VR_i|}{|VR|}
\]

(where \( VR_i \) is the set of documents in \( VR \) containing \( x_i \)). In practice, we might wish to smooth these estimates, which we could do by adding \( \frac{1}{2} \) to both the count of \( R_i \) and to relevant documents not containing the term, giving:

\[
p_i = \frac{|VR_i| + 0.5}{VR + 1}
\]

To allow for the judged sets being small, it is better to combine the new information with the original guess in a process of Bayesian updating. In this case we have:

\[
p_i^{(t+1)} = \frac{|VR_i| + \kappa p_i^{(t)}}{|VR| + \kappa}
\]

Here \( p_i^{(t)} \) is the \( t^{th} \) estimate for \( p_i \) in an iterative updating process and \( \kappa \) is the weight given to the Bayesian prior. In the absence of other evidence (and assuming that the user is perhaps indicating roughly 5 relevant or
non-relevant documents) then a value of around 4 is perhaps appropriate. That is, the prior is strongly weighted so that the estimate does not change too much from the evidence provided by a very small number of documents.

4. Repeat the above process, generating a succession of approximations to $R$ and hence $p_i$, until the user is satisfied.

It is also straightforward to derive a pseudo-relevance feedback version of this algorithm:

1. Assume that $p_i$ is constant over all $x_i$ in the query (as above).

2. Determine a guess for the size of the relevant document set. If unsure, a conservative (too small) guess is likely to be best. This motivates use of a fixed size set $V$ of highest ranked documents.

3. Improve our guesses for $p_i$ and $u_i$. We use the same methods as before for re-estimating $p_i$. If we simply use the distribution of $x_i$ in the documents in $V$ and let $V_i$ be the set of documents containing $x_i$ then we have:

$$p_i = \frac{|V_i|}{|V|}$$

and if we assume that documents that are not retrieved are not relevant then we have:

$$u_i = \frac{n_i - |V_i|}{N - |V|}$$

4. Go to step 2 until the ranking of the returned results converges.

Note that once we have a real estimate for $p_i$ then the $c_i$ weights used in the RSV value look almost like a tf-idf value. For instance, using some of the ideas we saw above, we have:

$$c_i = \log \left[ \frac{|V_i|}{|V|} \cdot \frac{1 - u_i}{u_i} \right] \approx \log \left[ \frac{|V_i|}{|V| - |V_i|} \cdot \frac{N}{n} \right]$$

But things aren’t quite the same: the first term measures the percent of (estimated) relevant documents that the term occurs in and not term frequency, and if we divided the two terms apart using log identities, we would be adding the two log terms rather than multiplying them.

11.3.5 PRP and BIM

Getting reasonable approximations of the needed probabilities for a probabilistic IR model is possible. It requires some restrictive assumptions. In the BIM these are:
A general problem seems to be that probabilistic models either require partial relevance information or else only allow one to derive apparently inferior term weighting models. However, some of these assumptions can be removed. For example, we can remove the assumption that index terms are independent. This is very far from true in practice. The canonical case being pairs like Hong and Kong which are strongly dependent, but dependencies can be complex such as when the term New enters into various dependencies with other words such as York, England, City, Stock and Exchange, and University. van Rijsbergen (1979) proposed a simple, plausible model which allowed a tree structure of dependencies, as in Figure 11.1. In this model each term can be directly dependent on one other. When it was invented in the 1970s, estimation problems held back the practical success of this model, but the idea was reinvented as the
Tree Augmented Naive Bayes model by Friedman and Goldszmidt (1996), who used it with some success on various machine learning data sets.

11.4 An appraisal and some extensions

Probabilistic methods are one of the oldest formal models in IR. Even in the 1970s they were held out as an opportunity to place IR on a former theoretical footing, and with the resurgence of probabilistic methods in the 1990s, that hope has returned, and probabilistic methods are again one of the currently hottest topics in IR. Traditionally, probabilistic IR has had neat ideas but the methods have never won on performance. At the end of the day, things also are not that different: one builds an information retrieval scheme in the exact same way as we have been discussing, it’s just that at the end, you score queries not by cosine similarity and tf-idf in a vector space, but by a slightly different formula motivated by probability theory. Indeed, sometimes people have adopted term weighting formulae from probabilistic models and used them in vector space (cosine calculation) retrieval engines. Here we briefly mention two influential extensions of the traditional probabilistic model, and in the next chapter, we look at the somewhat different probabilistic language modeling approach to IR.

11.4.1 Okapi BM25: a non-binary model

The BIM works reasonably in the contexts for which it was first designed (short catalog records and abstracts, of fairly consistent length), but for other search collections, it seems clear that a model should pay attention to term frequency and document length. The BM25 weights were developed as a way of building a probabilistic model sensitive to these quantities while not introducing too many additional parameters into the model (Spärck Jones et al. 2000). We will not develop the theory behind the model here, but just present a series of forms that build up to the now-standard form used for document scoring (there have been some variants). The simplest score for document \( j \) is just idf weighting of the query terms present:

\[
RSV_j = \sum_{i \in q} \log \frac{N}{n} \tag{11.24}
\]

We can improve on that by factoring in term frequency and document length:

\[
RSV_j = \sum_{i \in q} \left[ \log \frac{N}{n} \cdot \frac{(k_1 + 1)tf_{ij}}{k_1((1-b) + b \times (dl_j/\text{avdl})) + tf_{ij}} \right] \tag{11.25}
\]
If the query is long, then we might also put in similar weighting for query terms (this is appropriate if the queries are paragraph long information needs, but unnecessary for short queries):

\[
RSV_j = \sum_{i \in q} \left( \log \frac{N}{n} \cdot \frac{(k_1 + 1)tf_{ij}}{k_1((1 - b) + b \times (dl_j/avdl)) + tf_{ij}} \cdot \frac{(k_3 + 1)tf_{iq}}{k_3 + tf_{iq}} \right) \tag{11.26}
\]

Finally, if we have available relevance judgements, then we can use the full form of Equation (11.20) in place of the left hand side term (which is what we saw in Equation (11.21)):

\[
RSV_j = \sum_{i \in q} \left[ \log \left( \frac{(|VR| + \frac{1}{2})/(|VR| - |VR| + \frac{1}{2})}{(n - |VR| + \frac{1}{2})/(N - n - |VR| + |VR| + \frac{1}{2})} \right) \cdot \frac{(k_1 + 1)tf_{ij}}{k_1((1 - b) + b(dl_j/avdl)) + tf_{ij}} \cdot \frac{(k_3 + 1)tf_{iq}}{k_3 + tf_{iq}} \right] \tag{11.27}
\]

Here, \( N, n, |VR| \) and \( VR \) are used as in Sections 11.3.2 and 11.3.4, \( tf_{ij} \) and \( tf_{iq} \) are the frequency of the \( i \)th term in the document and query respectively, and \( dl_j \) and \( avdl \) are the document length and average document length of the whole collection. The first term reflects relevance feedback (or just idf weighting if no relevance information is available), the second term implements document term frequency and document length scaling, and the third term considers term frequency in the query (and is only necessary if the query is long). \( k_1 \) and \( k_3 \) are positive tuning parameters that scale the term frequency scaling inside the document and query respectively. A value of 0 corresponds to a binary model (no term frequency), and a large value corresponds to using raw term frequency. \( b \) is another tuning parameter \((0 \leq b \leq 1)\) which determines the scaling by document length: \( b = 1 \) corresponds to fully scaling the term weight by the document length, while \( b = 0 \) corresponds to no length normalization. There is no length normalization of queries (it is as if \( b = 0 \) for this term). The tuning parameters should ideally be set to optimize performance on a development query collection. In the absence of such optimization, reasonable values are to set \( k_1 = k_3 = 2 \) and \( b = 0.75 \). Relevance feedback can also involve augmenting the query (automatically or with manual review) with some (10–20) of the top terms in the known-relevant documents as ordered by the relevance factor \( \hat{c}_i \) from Equation (11.20). This BM25 term weighting formula has been used quite widely and quite successfully. See Spärck Jones et al. (2000) for extensive motivation and discussion of experimental results.

### 11.4.2 Bayesian network approaches to IR

Turtle and Croft (1989; 1991) introduced the use of Bayesian networks (Jensen
and Jensen 2001), a form of probabilistic graphical model for information retrieval. We skip the details because fully introducing the formalism of Bayesian networks would require much too much space, but conceptually, Bayesian networks use directed graphs to show probabilistic dependencies between variables, as in Figure 11.1, and have led to the development of sophisticated algorithms for propagating influence so as to allow learning and inference with arbitrary knowledge of arbitrary directed acyclic graphs. Turtle and Croft used a sophisticated network model to better model the complex dependencies between a document and a user’s information need.

The model decomposes into two parts: a document collection network and a query network. The document collection network is large, but can be precomputed: it maps from documents to terms to concepts. The concepts are a thesaurus-based expansion of the terms appearing in the document. The query network is relatively small but a new network needs to be built each time a query comes in, and then attached to the document network. The query network maps from query terms, to query subexpressions (built using probabilistic or “noisy” versions of AND and OR operators), to the user’s information need.

The result is a flexible probabilistic network which can generalize various simpler Boolean and probabilistic models. The system allowed efficient large-scale retrieval, and was the basis of the InQuery text retrieval system, used at the University of Massachusetts, and for a time sold commercially. On the other hand, the model still used various approximations and independence assumptions to make parameter estimation and computation possible. We would note that this model was actually built very early on in the modern era of using Bayesian networks, and there have been many subsequent developments in the theory, and the time is perhaps right for a new generation of Bayesian network-based information retrieval systems.

11.5 References and further reading

Longer introductions to the needed probability theory can be found in most introductory probability and statistics books, such as Grinstead and Snell (1997). An introduction to Bayesian utility theory can be found in Ripley (1996).

The probabilistic approach to IR originated in the UK in the 1950s. Robertson and Spärck Jones (1976b) introduces the main foundations of the BIM and van Rijsbergen (1979) presents in detail the classic BIM probabilistic model. The idea of the PRP is variously attributed to Stephen Robertson, M. E. Maron and William Cooper (the term “Probabilistic Ordering Principle” is used in Robertson and Spärck Jones (1976b), but PRP dominates in later work). Fuhr (1992) is a more recent presentation of probabilistic
IR, which includes coverage of other approaches such as probabilistic logics and Bayesian networks. Crestani et al. (1998) is another survey, though it adds little not in the earlier sources. Spärck Jones et al. (2000) is the definitive presentation of probabilistic IR experiments by the “London school”, and Robertson (2005) presents a retrospective on the group’s participation in TREC evaluations, including detailed discussion of the Okapi BM25 model and its development.

11.6 Exercises

Exercise 11.1
Think through the differences between standard vector space tf-idf weighting and the BIM probabilistic retrieval model on the first iteration.

Exercise 11.2
Think through the differences between vector space (pseudo) relevance feedback and probabilistic (pseudo) relevance feedback.

11.6.1 Okapi weighting

The so-called Okapi weighting, defined for a term $t$ and a document $d$ as follows, for positive constants $k_1$ and $b$:

\[
\text{OKAPI WEIGHTING} \quad \ln \frac{N - df_t + 0.5}{df_t + 0.5} \cdot \frac{(k_1 + 1)tf_{t,d}}{k_1(1 - b + b(\ell(d)/\ell_{\text{ave}})) + tf_{t,d}}.
\]

(11.28)

As before $N$ is the total number of documents; $\ell(d)$ is the length of $d$ and $\ell_{\text{ave}}$ is the average length of documents in the corpus. This weighting scheme has proven to work relatively well across a wide range of corpora and search tasks.