9 Language Modeling and Related Methods

This chapter presents alternative views of the ranking problem that are different from the probabilistic model introduced in Chapter 8. The retrieval methods presented in this chapter, although they have differing theoretical underpinnings, share several important characteristics. All may be viewed as ranking documents through some form of *language modeling*, in the sense that the actual occurrences of terms in a document are compared with the expected occurrences predicted from the characteristics of the collection and the document. The methods are also distinguished from the probabilistic model by their reduced emphasis on relevance, which may not be explicitly considered.

Although the methods presented in this chapter may be viewed as language modeling approaches in a broad sense, Sections 9.1 to 9.4 focus on the foundations of what is generally termed the "language modeling approach" in a narrower sense. This approach — established through the work of Berger, Croft, Hiemstra, Lafferty, Ponte, Zhai, and others — is characterized by its use of a document to construct a *generative model* for queries (Ponte and Croft, 1998; Hiemstra, 2001; Song and Croft, 1999; Miller et al., 1999; Berger and Lafferty, 1999; Lafferty and Zhai, 2001; Zhai and Lafferty, 2001, 2004). Over a short span of time, approximately 1998 to 2001, this language modeling approach grew from its roots in a SIGIR paper by Ponte and Croft (1998) into a leading framework for new IR research (Croft and Lafferty, 2003). Whenever references are made to "the language modeling approach" in the research literature, this narrower meaning is often intended.

Section 9.1 takes the Probability Ranking Principle as its starting point to derive and justify this generative query model. Given a document that is assumed to be relevant, we develop a language model from the document to estimate $p(q \mid d)$, the probability that query q would be entered to retrieve document d, and rank documents accordingly. We introduced the idea of language modeling in Chapter 1 when we examined term frequencies and developed the idea further in the context of data compression in Chapter 6. In the present chapter our language models are based on the simplest of those introduced in Chapter 1 — we work only with zero-order models derived from document and collection statistics.

Section 9.2 discusses the details of constructing a language model from a document, with particular emphasis on the smoothing process through which we assign nonzero probabilities to terms that do not appear within the document. It is important to assign positive probabilities to these terms because a relevant document may not contain every query term. In Section 9.3 we apply our smoothed language models to the ranking problem and develop specific ranking formulae by combining results from Sections 9.1 and 9.2. Section 9.4 considers an alternative theoretical foundation for the language modeling approach based on *Kullback-Leibler divergence*,

a method for determining the difference between two probability distributions. Building on this alternative foundation, we outline a method for query expansion under the language modeling approach.

Later sections consider a broader interpretation of language modeling. In Section 9.5 we introduce a retrieval method known as *divergence from randomness* (DFR), which explicitly compares a model for distributing terms at random with the actual distribution of these terms within documents to be ranked. Section 9.6 presents an approach to passage retrieval, the foundations of which are closely related to DFR and other language modeling approaches. Section 9.7 provides a comparative evaluation of several of the retrieval methods presented in the chapter and in previous chapters. Section 9.8 lists a number of emerging retrieval methods derived from or related to language modeling.

9.1 Generating Queries from Documents

As we did in Chapter 8, we take as our starting point the Probability Ranking Principle, as embodied in Equation 8.8:

$$\log \frac{p(r \mid D, Q)}{1 - p(r \mid D, Q)} = \log \frac{p(r \mid D, Q)}{p(\bar{r} \mid D, Q)} \tag{9.1}$$

$$= \log \frac{p(D, Q \mid r)\ p(r)}{p(D, Q \mid \bar{r})\ p(\bar{r})}. \tag{9.2}$$

In Equation 8.10 we expanded the joint probabilities appearing in this equation using the equality $p(D, Q \mid R) = p(D \mid Q, R) \cdot p(Q \mid R)$. In this section we develop the equation in the opposite direction, expanding the joint probabilities using the equality $p(D, Q \mid R) = p(Q \mid D, R) \cdot p(D \mid R)$.

$$\log \frac{p(r \mid D, Q)}{p(\bar{r} \mid D, Q)} = \log \frac{p(D, Q \mid r)\ p(r)}{p(D, Q \mid \bar{r})\ p(\bar{r})} \tag{9.3}$$

$$= \log \frac{p(Q \mid D, r)\ p(D \mid r)\ p(r)}{p(Q \mid D, \bar{r})\ p(D \mid \bar{r})\ p(\bar{r})} \tag{9.4}$$

$$= \log \frac{p(Q \mid D, r)\ p(r \mid D)}{p(Q \mid D, \bar{r})\ p(\bar{r} \mid D)} \tag{9.5}$$

$$= \log p(Q \mid D, r) - \log p(Q \mid D, \bar{r}) + \log \frac{p(r \mid D)}{p(\bar{r} \mid D)} \tag{9.6}$$

$$= \log p(Q \mid D, r) - \log p(Q \mid D, \bar{r}) + \text{logit}(p(r \mid D)) \tag{9.7}$$

We examine each of the probabilities in this last formula in detail.

For a given query q and a relevant document d, the value $p(Q = q \mid D = d, r)$ represents the probability that the user would enter the query q in order to retrieve d. In essence, conditioning on d provides us with an example of a relevant document. From this example we may then estimate the probability of a user entering q if a document such as d is desired. In particular, if a term appears in d with much greater frequency than random chance suggests, we might expect it to have a higher probability of appearing in the query.

On the other hand, consider the role of d in the probability $p(Q = q \mid D = d, \bar{r})$ in Equation 9.7. Here conditioning on d provides us with a weaker picture of the user's requirements. We have an example of what is not relevant but still can only guess about what is relevant. As a result it may be reasonable to assume that this probability is independent of d, that is $p(Q = q \mid D = d, \bar{r})$ is a constant. If we accept this assumption, we can drop the constant, leaving us with the rank-equivalent formula

$$\log p(Q \mid D, r) + \text{logit}(p(r \mid D)). \tag{9.8}$$

The probability $p(r \mid D)$ in the second term is independent of q and is the same for all queries. This prior probability of relevance might be interpreted as indicating something about the quality or importance of the document. In the context of Web search, for example, the home page of a university might be assigned a higher prior probability than the personal home page of a student attending the university (see Section 15.3). In the context of file system search, e-mail addressed to you personally might be assigned a higher value than e-mail you receive from a mailing list.

Nonetheless, in the absence of information suggested by the context of the application, the prior probability of relevance $p(r \mid D)$ is often assumed to be the same for all documents. Again, if we accept this assumption, we may drop this constant as an order-preserving transformation. Moreover, because exponentiation is also order-preserving, we may remove the logarithm, which leaves us with the ranking formula

$$p(Q \mid D, r). \tag{9.9}$$

If we wish, conditioning on relevance can be made implicit, thus reducing the equation to

$$p(Q = q \mid D = d). \tag{9.10}$$

This very simple formula plays the same role in relation to the language modeling approach as Equation 8.13 does in relation to the probabilistic retrieval model of Chapter 8. Given a document d and a query q, we score d for ranking purposes by estimating the probability $p(q \mid d)$.

In order to estimate this probability, the document is often considered to provide a *model* for generating the query q. Imagine a user formulating a query to retrieve relevant documents. She might attempt to think of some terms that should appear often in these relevant documents but rarely in nonrelevant documents. In making this attempt the user forms an image in her mind of what a relevant document looks like — what terms might distinguish it from other

documents in the collection. She then selects a few of these terms and enters them into a search engine. These terms become the query q.

To score d we assume it matches the image in the user's mind and estimate the probability that the user would select q to retrieve it. It is because of this assumed link between the user's information need — the image in her mind — and the document d that we can view d as providing a *generative model* for q.

Some IR researchers justify Equation 9.10 more directly through a different reasoning process. They simply state that we are interested in $p(Q, D)$, the joint distribution of queries and documents. The issue of relevance is ignored. Documents are ranked by

$$p(Q, D) = p(Q \mid D) \cdot p(D),$$

or equivalently by

$$\log p(Q \mid D) + \log p(D). \tag{9.11}$$

Similarly to how we handled the second term in Equation 9.8, we may interpret $\log p(D)$ as indicating the quality or importance of the document and treat it as a constant in the absence of other information. Note, however, the difference in this second term between Equations 9.11 and 9.8.

9.2 Language Models and Smoothing

In Chapter 1 we examined term distributions in the Shakespeare and other collections and introduced basic language modeling approaches. In addition, in the context of text compression in Chapter 6, we further explored the idea of finite-context models over a set of symbols. For predictive purposes we can imagine reading the unseen text of a document from left to right, using the language models to guess what will come next and assigning a probability to each possible symbol. If we do a good job of guessing, we are able to encode the document in fewer bits.

For retrieval our goal is slightly different. We must imagine that we have an example of a relevant document and ask: *What is the probability that term t would be entered by a user in order to retrieve a document such as this one?*

To answer this question we use the document to create a language model for the queries that may be entered to retrieve it. Our goal is to construct a document language model $\mathcal{M}_d(t)$ from document d. The simplest document language model is the maximum likelihood model $\mathcal{M}_d^{\text{ml}}(t)$ based on the counts of the terms appearing in the document:

$$\mathcal{M}_d^{\text{ml}}(t) = \frac{f_{t,d}}{l_d}. \tag{9.12}$$

Here, as usual, $f_{t,d}$ is the number of times t appears in d and l_d is the length of the document. Thus, for terms not appearing in the document $\mathcal{M}_d^{\text{ml}}(t) = 0$. Because $\mathcal{M}_d^{\text{ml}}(t)$ is a probability we have, as required,

$$\sum_{t \in \mathcal{V}} \mathcal{M}_d^{\text{ml}}(t) = \sum_{t \in d} \mathcal{M}_d^{\text{ml}}(t) = \sum_{t \in d} f_{t,d}/l_d = l_d/l_d = 1. \tag{9.13}$$

For example, the term "lord" appears 624 times in *Hamlet*, which has length 43,314, but only 78 times in *Macbeth*, which has length 26,807. Thus, a language model based on *Hamlet* predicts a much higher probability for that term than one based on *Macbeth*.

$$\mathcal{M}^{\text{ml}}_{\text{Hamlet}}(\text{"lord"}) = \frac{f_{\text{lord,Hamlet}}}{l_{\text{Hamlet}}} = \frac{624}{43,314} \approx 1.441\%$$

$$\mathcal{M}^{\text{ml}}_{\text{Macbeth}}(\text{"lord"}) = \frac{f_{\text{lord,Macbeth}}}{l_{\text{Macbeth}}} = \frac{78}{26,807} \approx 0.291\%$$

On the other hand, *Hamlet* predicts a lower probability for the term "lady" because it appears 30 times in *Hamlet* but 196 times in *Macbeth*.

$$\mathcal{M}^{\text{ml}}_{\text{Hamlet}}(\text{"lady"}) = \frac{f_{\text{lady,Hamlet}}}{l_{\text{Hamlet}}} = \frac{30}{43,314} \approx 0.069\%$$

$$\mathcal{M}^{\text{ml}}_{\text{Macbeth}}(\text{"lady"}) = \frac{f_{\text{lady,Macbeth}}}{l_{\text{Macbeth}}} = \frac{196}{26,807} \approx 0.731\%$$

Because $\mathcal{M}_d^{\text{ml}}(t)$ is nothing more than term frequency scaled by document length, we might suspect that by itself it may not be sufficient to compute estimates of $p(q \mid d)$ that will provide a satisfactory document ranking. In particular, given that d is just a single example of a relevant document and that d may consist of only a few hundred words, the estimates it provides have the potential to be wildly inaccurate. Moreover, the model assigns a probability of 0 to all terms not appearing in the document, implying that it is impossible for these terms to appear in the query.

To address these problems when using language models based on documents or examples of similar size, it is common in information retrieval, as well as in other areas such as speech recognition, to *smooth* these models using a *background language model* in the hope of improving accuracy. In information retrieval the collection as a whole provides a convenient basis for this background model.

We define $\mathcal{M}_C(t)$ as the maximum likelihood language model based on term frequencies in the collection as a whole:

$$\mathcal{M}_C(t) = l_t/l_C, \tag{9.14}$$

where l_t is the number of times t occurs in the collection C and l_C is the total number of tokens in the collection.

$$\begin{aligned} M_C(\text{"lord"}) &= \frac{l_{\text{lord}}}{l_C} = \frac{3,346}{1,271,504} \approx 0.263\% \\ M_C(\text{"lady"}) &= \frac{l_{\text{lady}}}{l_C} = \frac{1,031}{1,271,504} \approx 0.081\% \end{aligned}$$

Unlike the examples in Chapter 1, in this example l_C includes all tokens, tags as well as words.

We now consider two well-known smoothing methods. The first method, known as *Jelinek-Mercer smoothing* (Jelinek and Mercer, 1980; Chen and Goodman, 1998), is a simple linear combination of the document language model and the collection language model,

$$M_d^\lambda(t) = (1-\lambda) \cdot M_d^{\text{ml}}(t) + \lambda \cdot M_C(t), \tag{9.15}$$

where λ is a parameter with a value between 0 and 1 that controls the relative weight given to the document and collection language models. For example, if $\lambda = 0.5$,

$$\begin{aligned} M_{\text{Hamlet}}^\lambda(\text{"lord"}) &= (1-\lambda) \cdot M_{\text{Hamlet}}^{\text{ml}}(\text{"lord"}) + \lambda \cdot M_C(\text{"lord"}) \\ &= 0.5 \cdot \frac{78}{26,807} + 0.5 \cdot \frac{3,346}{1,271,504} \approx 0.277\%. \end{aligned}$$

Because $M_d^\lambda(t)$ represents a probability, we have, as required,

$$\sum_{t \in V} (1-\lambda) \cdot M_d^{\text{ml}}(t) + \lambda \cdot M_C(t) = (1-\lambda) \cdot \sum_{t \in V} M_d^{\text{ml}}(t) + \lambda \cdot \sum_{t \in V} M_C(t) = 1. \tag{9.16}$$

The intuition underlying our second smoothing method is to pretend we have added an extra $\mu > 0$ tokens to each document in the collection and have distributed them according to the collection language model $M_C(t)$. For example, if $\mu = 1,000$, we would conceptually add $\mu \cdot M_C(\text{"lord"}) = 2.6315$ occurrences of the term "lord" to every document. Obviously, in reality it is not possible to add a fractional number of terms to a document, but it works as a mathematical trick. The maximum likelihood language model based on these new documents would then be

$$M_d^\mu(t) = \frac{f_{t,d} + \mu M_C(t)}{l_d + \mu},$$

where $f_{t,d}$ is the number of times t appears in the original document and l_d is the original length. The impact of the additional terms depends on the length of the document. The longer the document, the lower the impact, with values approaching $M_d(t)$ in the limit. This smoothing method is known as *Dirichlet smoothing* (Chen and Goodman, 1998) because it can be derived from a Dirichlet distribution with appropriate parameters.

If a term t does not appear in a document d, then $\mathcal{M}_d^{\lambda}(t) = \lambda \cdot \mathcal{M}_C(t)$ and $\mathcal{M}_d^{\mu}(t) = (\mu/(l_d + \mu)) \cdot \mathcal{M}_C(t)$. In both cases the smoothed models assign a probability that is a constant factor of the collection probability, and thus the relative probability assigned to terms not appearing in the document remains the same.

9.3 Ranking with Language Models

Equation 9.10 suggests that we are ranking documents according to the probability that q will be entered as a query, given that d is an example of a relevant document. We are now ready to apply the smoothed language models of Section 9.2 to the problem of estimating this probability. We start by applying independence assumptions similar to Assumption Q on page 262, thereby reducing the problem to that of estimating probabilities for individual terms in the query. Given a query vector $q = \langle t_1, t_2, ..., t_n \rangle$ and a document d, we may estimate $p(q \mid d)$ as

$$p(q \mid d) = p(|q| = n) \cdot \prod_{i=1}^{n} p(t_i \mid d), \tag{9.17}$$

where $p(|q| = n)$ represents the probability that the user would enter a query of length n. In order to estimate the probability that the user will enter q, we must estimate this probability for the query length — assuming that this length is independent of the document and query terms — as well as a probability for each individual term — assuming that the selection of one term is independent of the selection of the others.

Fortunately, it is safe to ignore the query length. Because the query is fixed, it is sufficient to consider the probability that the user would enter q out of all possible queries of length n rather than out of all possible queries of any length. In Section 9.4, where we briefly consider query expansion under the language modeling approach, the need to estimate query length will reappear. For now, however, we estimate $p(q \mid d)$ as

$$p(q \mid d) = \prod_{i=1}^{n} p(t_i \mid d). \tag{9.18}$$

Because the order of terms in the query vector is of no significance in this equation, we may treat q as a set and make the query-term frequency explicit in the equation:

$$p(q \mid d) = \prod_{t \in q} p(t \mid d)^{q_t}. \tag{9.19}$$

A document language model may then be used to estimate a value for each $p(t\,|\,d)$, and thus we would rank documents according to

$$p(q\,|\,d) \;=\; \prod_{t\in q} \mathcal{M}_d(t)^{q_t}, \tag{9.20}$$

where $\mathcal{M}_d(t)$ may be $\mathcal{M}_d^{\lambda}(t)$, $\mathcal{M}_d^{\mu}(t)$, or another document language model. If we substitute $\mathcal{M}_d^{\lambda}(t)$ for $\mathcal{M}_d(t)$, we get

$$p(q\,|\,d) \;=\; \prod_{t\in q} \left((1-\lambda)\cdot \mathcal{M}_d^{\mathrm{ml}}(t) + \lambda\cdot \mathcal{M}_C(t)\right)^{q_t}. \tag{9.21}$$

If we substitute $\mathcal{M}_d^{\mu}(t)$ for $\mathcal{M}_d(t)$, we get

$$p(q\,|\,d) \;=\; \prod_{t\in q} \left(\frac{f_{t,d}+\mu\mathcal{M}_C(t)}{l_d+\mu}\right)^{q_t}. \tag{9.22}$$

We could stop at this point and use either Equation 9.21 or Equation 9.22 for ranking. However, it is valuable to expend a little extra effort over the next few pages in order to achieve some additional insight into the language modeling approach and its relationship to other methods. You may have noticed that many of the characteristics saw in the ranking formulae of Chapters 2 and 8 are absent from these equations. Unlike the previous formulae, these equations do not take a TF-IDF form. Moreover, the number of documents in the collection (N) and the number of documents containing the term (N_t) do not appear in the equations at all. Nonetheless, with a little work we can find something close to TF-IDF hidden in these equations while simplifying them at the same time.

For convenience as well as consistency with the other approaches, we work with logarithms from this point forward rather than directly with the probabilities. First, we consider the terms contained in the document separately from the terms not contained in the document.

$$\log p(q\,|\,d) \;=\; \sum_{t\in q} q_t\cdot \log p(t\,|\,d) \tag{9.23}$$

$$\;=\; \sum_{t\in q\cap d} q_t\cdot \log p(t\,|\,d) + \sum_{t\in q\setminus d} q_t\cdot \log p(t\,|\,d) \tag{9.24}$$

When t is contained in the document, we use $\mathcal{M}_d(t)$ to estimate $p(t\,|\,d)$, where $\mathcal{M}_d(t)$ may be either $\mathcal{M}_d^{\lambda}(t)$ or $\mathcal{M}_d^{\mu}(t)$. When t is not contained in the document, $\mathcal{M}_d(t)$ takes the form $\alpha_d\mathcal{M}_C(t)$, where α_d is a factor depending only on characteristics of d (and not on q). As indicated at the end of Section 9.2, $\alpha_d = \lambda$ for $\mathcal{M}_d^{\lambda}(t)$ and $\alpha_d = \mu/(l_d+\mu)$ for $\mathcal{M}_d^{\mu}(t)$.

We may now substitute $\mathcal{M}_d(t)$ and $\alpha_d \cdot \mathcal{M}_C(t)$ into Equation 9.24, and with a little rearrangement, we eliminate summation over the query terms not appearing in the document:

$$\log p(q|d) = \sum_{t \in q \cap d} q_t \cdot \log \mathcal{M}_d(t) + \sum_{t \in q \setminus d} q_t \cdot \log(\alpha_d \cdot \mathcal{M}_C(t)) \tag{9.25}$$

$$= \sum_{t \in q \cap d} q_t \cdot \log \mathcal{M}_d(t) - \sum_{t \in q \cap d} q_t \cdot \log(\alpha_d \cdot \mathcal{M}_C(t)) \tag{9.26}$$

$$+ \sum_{t \in q \cap d} q_t \cdot \log(\alpha_d \cdot \mathcal{M}_C(t)) + \sum_{t \in q \setminus d} q_t \cdot \log(\alpha_d \cdot \mathcal{M}_C(t)) \tag{9.27}$$

$$= \sum_{t \in q \cap d} q_t \cdot \log \frac{\mathcal{M}_d(t)}{\alpha_d \mathcal{M}_C(t)} + \sum_{t \in q} q_t \cdot \log(\alpha_d \cdot \mathcal{M}_C(t)) \tag{9.28}$$

$$= \sum_{t \in q \cap d} q_t \cdot \log \frac{\mathcal{M}_d(t)}{\alpha_d \mathcal{M}_C(t)} + n \cdot \log \alpha_d + \sum_{t \in q} q_t \cdot \log \mathcal{M}_C(t) . \tag{9.29}$$

Here $n = \sum_{t \in q} q_t$ represents the number of tokens in the query. The last term, $\sum_{t \in q} q_t \cdot \log \mathcal{M}_C(t)$, is constant for all documents and may be dropped to give the rank-equivalent formula

$$\sum_{t \in q \cap d} q_t \cdot \log \frac{\mathcal{M}_d(t)}{\alpha_d \mathcal{M}_C(t)} + n \cdot \log \alpha_d . \tag{9.30}$$

This formula has two parts. The part on the left is a sum over weights associated with the query terms appearing in the document and is reminiscent of the ranking formulae presented in previous chapters. The part on the right may be viewed as a document-specific adjustment or normalization that is independent of the specific query terms but not of query or document length.

We now specialize Equation 9.30 by replacing $\mathcal{M}_d(t)$ with each of our two smoothed language models, $\mathcal{M}_d^{\lambda}(t)$ and $\mathcal{M}_d^{\mu}(t)$. Substituting $\mathcal{M}_d^{\lambda}(t)$ gives

$$\sum_{t \in q \cap d} q_t \cdot \log \frac{\mathcal{M}_d^{\lambda}(t)}{\alpha_d \mathcal{M}_C(t)} + n \cdot \log \alpha_d = \sum_{t \in q \cap d} q_t \cdot \log \frac{(1-\lambda)\mathcal{M}_d^{\text{ml}}(t) + \lambda \mathcal{M}_C(t)}{\lambda \mathcal{M}_C(t)} + n \cdot \log \lambda$$

$$= \sum_{t \in q \cap d} q_t \cdot \log \frac{(1-\lambda) f_{t,d}/l_d + \lambda l_t/l_C}{\lambda l_t/l_C} + n \cdot \log \lambda .$$

The term $n \cdot \log \lambda$ is a constant and may be dropped. A little re-arranging gives the final form of the ranking formula:

$$\sum_{t \in q} q_t \cdot \log \left(1 + \frac{1-\lambda}{\lambda} \cdot \frac{f_{t,d}}{l_d} \cdot \frac{l_C}{l_t}\right). \tag{9.31}$$

In the remainder of the book we refer to this formula as *language modeling with Jelinek-Mercer smoothing* (LMJM). In general the optimal value for λ appears to be related to query length, with larger values being more appropriate for longer queries. In the absence of training data or other information, experience has shown $\lambda = 0.5$ to be an acceptable value. Unless otherwise indicated, we use $\lambda = 0.5$ for examples and experiments.

We return to consider the document collection in Table 2.1. Given the query ⟨"quarrel","sir"⟩, we would compute the LMJM score for document 1 as follows:

$$\begin{aligned}&\log\left(1+\tfrac{1-\lambda}{\lambda}\cdot\frac{f_{\text{quarrel},1}}{l_1}\cdot\frac{l_C}{l_{\text{quarrel}}}\right)+\log\left(1+\tfrac{1-\lambda}{\lambda}\cdot\frac{f_{\text{sir},1}}{l_1}\cdot\frac{l_C}{l_{\text{sir}}}\right)\\ &= \log\left(1+\tfrac{0.5}{0.5}\cdot\tfrac{1}{4}\cdot\tfrac{28}{2}\right)+\log\left(1+\tfrac{0.5}{0.5}\cdot\tfrac{1}{4}\cdot\tfrac{28}{5}\right) \approx 3.43.\end{aligned}$$

Before proceeding, it is worthwhile to examine briefly the structure of Equation 9.31. Document-oriented collection statistics, such as N and N_t, are nowhere to be seen. However, the presence of the collection language model, l_C/l_t, inside the logarithm is suggestive of IDF and its close association with $f_{t,d}$ is suggestive of TF-IDF, so something of the equations of Chapters 2 and 8 can be seen here.

The document length plays an interesting role. The fraction $f_{t,d}/l_d$ is the average number of occurrences of t per document token. When it is multiplied by l_C/l_t, the result may be interpreted as the ratio of the actual number of occurrences per token to the expected number of occurrences based on the collection language model.

We now consider Dirichlet smoothing. Substituting $\mathcal{M}_d^\mu(t)$ for $\mathcal{M}_d(t)$ in Equation 9.30 and simplifying gives the ranking formula for $\mu > 0$,

$$\sum_{t\in q\cap d} q_t\cdot\log\frac{\mathcal{M}_d^\mu(t)}{\alpha_d\mathcal{M}_C(t)}+n\cdot\log\alpha_d = \sum_{t\in q} q_t\cdot\log\left(1+\frac{f_{t,d}}{\mu}\cdot\frac{l_C}{l_t}\right)-n\cdot\log\left(1+\frac{l_d}{\mu}\right), \quad (9.32)$$

which we refer to as *language modeling with Dirichlet smoothing* (LMD). In this formula the document length normalization is separated out into the term on the right. As in Equation 9.31, you may detect a hint of TF-IDF in the relationship between term frequency and the collection language model. Unless otherwise indicated we use this formula for experiments and examples with $\mu = 1000$. We emphasize, however, that in practice the value of μ should be tuned using appropriate training data (see Chapter 11).

If all documents in the collection have the same length, the two smoothing methods can be made equivalent by setting $\mu = l_d\cdot\frac{\lambda}{1-\lambda}$. This observation suggests that we might pick an acceptable value for μ, based on an existing value for λ, by replacing l_d with the average document length l_{avg}, resulting in the value $\mu = l_{avg}\cdot\frac{\lambda}{1-\lambda}$. If we treat 0.5 as the default value for λ, the corresponding default value for μ is then simply the average document length. By substituting l_{avg} for μ in Equation 9.32 and by noting that $l_{avg} = l_C/N$, the LMD ranking formula reduces to

$$\sum_{t\in q} q_t\cdot\log\left(1+f_{t,d}\cdot\frac{N}{l_t}\right)-n\cdot\log\left(1+\frac{l_d}{l_{avg}}\right). \quad (9.33)$$

The presence of N in this formula strengthens the relationship with IDF that we noted earlier. The equation also contains hints of the document length normalization seen in Okapi BM25.

If we use LMD to rank the collection in Table 2.1 with respect to the query ⟨"quarrel","sir"⟩, we would compute a score for document 1 as follows:

$$\begin{aligned}&\log\left(1+f_{\text{quarrel},1}\cdot\frac{N}{l_{\text{quarrel}}}\right)+\log\left(1+f_{\text{sir},1}\cdot\frac{N}{l_{\text{sir}}}\right)-n\cdot\log\left(1+\frac{l_1}{l_{avg}}\right)\\ &=\ \log\left(1+1\cdot\tfrac{5}{2}\right)+\log\left(1+1\cdot\tfrac{5}{5}\right)-2\cdot\log\left(1+\tfrac{4}{5.6}\right)\ \approx\ 1.25.\end{aligned}$$

9.4 Kullback-Leibler Divergence

An alternative theoretical framework for understanding and working with the language modeling approach is provided by a concept known as *Kullback-Leibler divergence*. KL divergence, also known as *relative entropy*, is a method for comparing two probability distributions.

Given continuous probability distributions $f(x)$ and $g(x)$ the KL divergence between them is defined as

$$\int_{-\infty}^{\infty} f(x)\cdot\log\frac{f(x)}{g(x)}\,dx\,. \tag{9.34}$$

In information retrieval, we normally work with discrete distributions, for which KL divergence takes the form

$$\sum_x f(x)\cdot\log\frac{f(x)}{g(x)}\,. \tag{9.35}$$

Larger values indicate greater divergence. When f and g represent the same distribution, their KL divergence is zero, since $\log\left(f(x)/g(x)\right)=\log 1=0$.

For example, the flip of a "fair" coin will produce heads or tails with equal probability. If an "unfair" coin produces heads with a 40% probability and tails with a 60% probability, the KL divergence between the fair and unfair coin may be computed as

$$0.5\cdot\log\frac{0.5}{0.4}+0.5\cdot\log\frac{0.5}{0.6}\ \approx\ 0.02945.$$

In this example, we choose 2 for the base of the logarithm because the choice is arbitrary.

KL divergence is not symmetric in the sense that reversing the roles of $f(x)$ and $g(x)$ may produce a different value. For example, if we reverse the roles of the fair and unfair coins, the KL divergence becomes

$$0.4\cdot\log\frac{0.4}{0.5}+0.6\cdot\log\frac{0.6}{0.5}\ =\ 0.02905.$$

Due to this asymmetry KL divergence is sometimes viewed as comparing a "true" distribution with another distribution in which $f(x)$ in Equation 9.35 represents this true distribution. From an information-theoretic standpoint, KL divergence indicates the average number of extra bits

per symbol needed to transmit or compress a message if we assume its symbols are distributed according to g instead of the true distribution f.

In order to apply KL divergence to ranking, we construct a language model from the query $M_q(t)$ in much the same way that we constructed a language model from the document. The simplest language model is the maximum likelihood model: the ratio of the number of times t appears in the query to the length of the query:

$$M_q^{\text{ml}}(t) = \frac{q_t}{n}. \tag{9.36}$$

It is also possible to create more complex query language models through smoothing or other processes, just as we did for the document language models.

We then apply KL divergence to rank documents by computing the divergence between the query language model and the document language model:

$$\sum_{t \in \mathcal{V}} M_q(t) \cdot \log \frac{M_q(t)}{M_d(t)} = \sum_{t \in \mathcal{V}} M_q(t) \cdot \log M_q(t) - \sum_{t \in \mathcal{V}} M_q(t) \cdot \log M_d(t). \tag{9.37}$$

The summation on the left is the same for all documents and may be dropped as an order-preserving transform. The summation on the right, without the negative sign, increases with decreasing divergence and is therefore suitable as a ranking formula:

$$\sum_{t \in \mathcal{V}} M_q(t) \cdot \log M_d(t). \tag{9.38}$$

Now, if we replace $M_q(t)$ with the maximum likelihood language model $M_q^{\text{ml}}(t)$, the formula becomes

$$\sum_{t \in \mathcal{V}} M_q^{\text{ml}}(t) \cdot \log M_d(t) = \frac{1}{n} \cdot \sum_{t \in q} q_t \cdot \log M_d(t). \tag{9.39}$$

The constant $\frac{1}{n}$ may be dropped to give the rank-equivalent formula

$$\sum_{t \in q} q_t \cdot \log M_d(t), \tag{9.40}$$

which is exactly Equation 9.20 (in log-space).

Instead of using a maximum likelihood model for $M_q(t)$ in Equation 9.38, we may instead view the presence of a query language model as an opportunity for query expansion, extending the query language model to estimate nonzero probabilities for terms not appearing in the original query. In this way we can assign positive scores to documents that may not contain any of the query terms. For example, given the query ("law","enforcement","dogs") in Figure 1.8 (page 25), a document discussing the use of police canine (or K-9) units for drug searches would certainly be relevant. As a result, performance might be improved by adding the terms "police", "canine", "K-9", "drug", and "searches" to the query, perhaps with appropriate weighting to

reflect their secondary status as expansion terms. The language model $\mathcal{M}_q(t)$ represents a bridge between the original query terms and potential expansion terms, thereby providing a theoretically sound route for their identification and weighting.

Lafferty and Zhai (2001) suggest an approach for expanding the query language model based on a random "walk" through the collection, starting at a document containing a query term. At the first step of the walk a random document containing a query term is selected in which the selection is weighted by term frequency and perhaps other factors. A random term is then selected from the new document according to $\mathcal{M}_d(t)$. With probability p_{stop} the walk stops. With probability $1 - p_{stop}$ it continues. We then randomly select a document containing the new term, then a new term from that document, and so on. The query language model $\mathcal{M}_q(t)$ is then based on the probability of stopping at term t during this walk.

Lafferty and Zhai present a matrix formalization of this informal idea, thus effectively allowing the walk to be performed for all terms simultaneously. For efficiency they stop the walk after a small number of steps in order to avoid assigning a tiny probability to a prohibitively large number of terms. They report significant effectiveness improvements over several collections, including TREC45.

9.5 Divergence from Randomness

The *divergence from randomness* (DFR) approach to information retrieval explicitly assumes a random process for the distribution of terms in documents, and then ranks documents by considering the probability that the actual term distribution found in a document would occur by chance. Similar considerations appear implicitly in the language modeling approach through its incorporation of a collection language model into the smoothing process. For example, we noted on page 295 that Equation 9.31 includes the ratio of the actual number of occurrences in a document to the expected number of occurrences based on the collection language model.

An important property of DFR is the absence of the seemingly arbitrary parameters that appear in other methods and require tuning over a training set. Parameters such as μ in LMD, λ in LMJM, and k_1 in BM25 often appear unintuitive, defying an easy explanation for their presence. DFR offers retrieval effectiveness comparable with these methods in a non-parametric form.

In this section we also revisit the notion of eliteness that we introduced in Section 8.4. Having determined the probability that a random document d would contain $f_{t,d}$ occurrences of term t where $f_{t,d} > 0$, we exploit the notion of eliteness to estimate the probability that the document is actually "about" the concept embodied in the term. In Section 8.4 this notion was modeled by a two-Poisson distribution. In this section we present an approach based on Laplace's *law of succession*.

At its most generic the core of the DFR approach may be summarized by the formula

$$(1 - P_2) \cdot (-\log P_1)\,, \tag{9.41}$$

In this formula P_1 represents the probability that a random document d contains exactly $f_{t,d}$ occurrences of t. The value $-\log P_1$ may be viewed as the number of bits of information, called the *self-information*, associated with d containing exactly $f_{t,d}$ occurrences of t (see Section 6.1). Our random process for distributing occurrences of the term t across the document collection is unlikely to assign a large proportion of these terms to the specific document d. Thus P_1 may decrease rapidly as $f_{t,d}$ increases.

P_2 provides an adjustment that reflects eliteness, thus correcting for this rapid decrease. If d is elite in the t, we might assume that the occurrences of t that appear within it are not accidental. Suppose we begin reading d, which is elite in t, in order to count the number of occurrences of t it contains. Well before we reach its end, we discover $f_{t,d} - 1$ occurrences of t. This discovery suggests that we should expect to find more occurrences and P_2 is the probability of finding at least one. P_2 increases as $f_{t,d}$ increases. Thus, $(1 - P_2)$ in Equation 9.41 decreases as $f_{t,d}$ increases.

Thus far we have considered only a single term. To rank documents with respect to a multi-term query, we make the usual independence assumption, giving the ranking formula

$$\sum_{t \in q} q_t \cdot (1 - P_2) \cdot (-\log P_1) . \tag{9.42}$$

In the remainder of the presentation we focus on estimating P_1 and P_2 for a given term t with the understanding that these estimates will be used in this formula for ranking.

The theory underlying the DFR was first developed by Amati and van Rijsbergen (2002), and was validated through experimentation at TREC conferences (Plachouras et al., 2002; Amati et al., 2003). In addition to providing substantial theoretical justification for Equation 9.41, Amati and van Rijsbergen (2002) present and evaluate seven methods for estimating P_1 and two methods for estimating P_2. In this chapter we examine in detail only one of each, chosen for their ability to illustrate the reasoning underlying the approach as well as their retrieval effectiveness.

In addition Amati and van Rijsbergen present two methods for incorporating document length normalization into the DFR approach. For the time being we will ignore this complicating issue by assuming that all documents are of the same length. Toward the end of our presentation we will return to the issue by handling document length normalization in the spirit of Equation 8.45, computing an adjusted term frequency for documents of nonaverage length and using it to replace the actual term frequency.

9.5.1 A Model of Randomness

Suppose we randomly distribute terms into documents. If we have l_t occurrences of term t distributed across N documents, then

$$f_{t,1} + f_{t,2} + \ldots + f_{t,N} = l_t , \tag{9.43}$$

where $f_{t,i}$ is the number of occurrences of term t in the ith document. How many different ways can l_t occurrences be distributed across N documents, assuming that the documents are indistinguishable? For example, four occurrences can be distributed across three documents in four different ways: (1) all occurrences into one document, (2) three occurrences into one document and one into another, (3) two occurrences in one document and one in each of the other documents, or (4) two occurrences in two documents. In other words, how many different arrangements of terms will satisfy Equation 9.43?

To answer this question, Amati and van Rijsbergen recognize that this problem is identical to the problem addressed by *Bose-Einstein statistics*, which computes the number of ways indistinguishable particles may be assigned to energy states in thermal equilibrium. The solution may be expressed as a binomial coefficient:

$$\begin{pmatrix} N + l_t - 1 \\ l_t \end{pmatrix} = \frac{(N + l_t - 1)!}{(N-1)!\, l_t!}. \tag{9.44}$$

Alternatively, the problem may be viewed as equivalent to the problem of determining the number of ways in which m balls can be distributed across n indistinguishable bins, a version of the *occupancy problem* from combinatorics.

In order to compute an estimate of P_1 for term t, we assume that a given document d is found to contain $f_{t,d}$ occurrences of t. A random distribution of the remaining $l_t - f_{t,d}$ occurrences into the remaining documents must satisfy the equation

$$f_{t,1} + \cdots + f_{t,d-1} + f_{t,d+1} + \cdots + f_{t,N} = l_t - f_{t,d}. \tag{9.45}$$

The number of ways to satisfy this equation follows from Equation 9.44:

$$\begin{pmatrix} (N-1) + (l_t - f_{t,d}) - 1 \\ l_t - f_{t,d} \end{pmatrix} = \frac{((N-1) + (l_t - f_{t,d}) - 1)!}{(N-2)!\, (l_t - f_{t,d})!}. \tag{9.46}$$

This equation assumes that a selected document contains $f_{t,d}$ occurrences and represents the number of ways the remaining terms can be distributed. Equation 9.44 represents the number of ways l_t occurrences may be distributed across N documents. Thus, the ratio of these equations represents P_1, the probability that a selected document contains $f_{t,d}$ occurrences:

$$P_1 = \frac{\begin{pmatrix} (N-1) + (l_t - f_{t,d}) - 1 \\ l_t - f_{t,d} \end{pmatrix}}{\begin{pmatrix} N + l_t - 1 \\ l_t \end{pmatrix}} = \frac{((N-1) + (l_t - f_{t,d}) - 1)!\, (N-1)!\, l_t!}{(N-2)!(l_t - f_{t,d})!\, (N + l_t - 1)!}. \tag{9.47}$$

Unfortunately, the presence of factorials in Equation 9.47 makes it difficult to work with it directly. Instead, Amati and van Rijsbergen provide two methods for estimating its value and

demonstrate that both methods provide similar performance in terms of effectiveness. The simpler of these methods provides an estimate for P_1 of

$$P_1 = \left(\frac{1}{1+l_t/N}\right)\left(\frac{l_t/N}{1+l_t/N}\right)^{f_{t,d}}, \tag{9.48}$$

and therefore

$$-\log P_1 = \log(1+l_t/N) + f_{t,d} \cdot \log(1+N/l_t). \tag{9.49}$$

The term on the right has a form reminiscent of TF-IDF, but with l_t taking the role of N_t.

9.5.2 Eliteness

Amati and van Rijsbergen obtain one estimate for P_2 by way of Laplace's law of succession, which can best be explained by an example. Suppose we have seen the sun rise on each of $m-1$ successive mornings. In the absence of other information, such as a physical model of the Earth orbiting the sun, what probability should we assign to the event that the sun will rise tomorrow? Even though we are fairly certain this event will happen, it is not appropriate to assign it a value of 1, absolute certainty. After all, the sun may not rise tomorrow. Instead (without getting into the mathematical details) the law of succession suggests the value $m/(m+1)$.

Amati and van Rijsbergen apply the law of succession to estimate

$$P_2 = \frac{f_{t,d}}{f_{t,d}+1}. \tag{9.50}$$

They explain this equation as the "conditional probability of having one more token of the term in the document [...], assuming that the length of a relevant document is very large". Substituting this estimate along with Equation 9.49 into 9.41 gives

$$(1-P_2)(-\log P_1) = \frac{\log(1+l_t/N) + f_{t,d}\log(1+N/l_t)}{f_{t,d}+1}. \tag{9.51}$$

The term-frequency component in this formula resembles the term-frequency component of the Okapi BM25 ranking formula (Equation 8.48 on page 272), and possesses a similar saturation property. Regardless of the value of $f_{t,d}$, the value of Equation 9.51 is bounded by $\log(1+l_t/N) + \log(1+N/l_t)$.

9.5.3 Document Length Normalization

Equation 9.51 assumes that all documents have the same length. When documents vary in length, Amati and van Rijsbergen suggest a normalization in which an adjusted term frequency $f'_{t,d}$ replaces $f_{t,d}$ in the equation. They derive and evaluate two methods for computing this

adjustment. The better-performing of these methods is

$$f'_{t,d} = f_{t,d} \cdot \log\left(1 + l_{avg}/l_d\right). \tag{9.52}$$

They call the combination of Equation 9.49 and 9.50, as adjusted by Equation 9.52, the GL2 variant of the DFR approach. We use this variant for the experiments reported in this book.

9.6 Passage Retrieval and Ranking

Most of the ranking methods we examine in this book rank documents. Depending on the application environment, these documents may correspond to Web pages, news articles, e-mail messages, or a combination of these and other document types. In some environments it may be appropriate to rank elements within documents, such as the pages in a book or the sections in a news article. In other circumstances *arbitrary passage retrieval* may be desirable so that any fragment of text within a document may be returned as a ranked result.

William **Shakespeare** — Wikipedia, the free encyclopedia

At the age of 18, **Shakespeare** married the 26-year-old Anne Hathaway. The consistory court of the Diocese of Worcester issued a **marriage** licence on 27 November 1582. Two of Hathaway's neighbours posted bonds the next day ...

en.wikipedia.org/wiki/William_**Shakespeare**

Figure 9.1 A typical search engine result for the query ⟨"shakespeare","marriage"⟩, evaluated against Wikipedia. The result provides a brief snippet from the document that shows the query terms in context.

One such circumstance is generation of snippets to provide a simple summary of the contents of a retrieved document for presentation to the user. Figure 9.1 provides an example of the result a search engine might return from Wikipedia given the query ⟨"shakespeare","marriage"⟩. Along with the title of the Web page and its URL, the search engine provides a snippet extracted from the body of the docnment to illustrate the context in which the search terms appear.

Question answering is another application in which arbitrary passage retrieval may be valuable (Tellex et al., 2003). Given a question such as "What is the population of India?", a question answering system attempts to provide an exact answer ("1.12 billion") rather than a document that may contain the answer. Passage retrieval is often an early step in question answering. Keywords are extracted from the question and formed into a query for processing by an IR system (⟨"population","india"⟩). Passages containing these terms are then retrieved from the corpus and analyzed to extract and validate possible answers. An interval containing the query terms in close proximity ("...population of India, ...") is likely to be part of a slightly longer

but still short passage containing the answer ("...The population of India is..."). The interval represents a "hot spot" within the collection near which the answer may be found.

In Section 2.2.2 we presented a simple ranking method based solely on term proximity. That method locates *covers* for a query vector $q = \langle t_1, t_2, ..., t_n \rangle$, where a cover is defined as an interval of text $[u, v]$ that contains at least one occurrence of each term in the query and does not contain a smaller interval that also contains all the terms.

We now extend the notion of a cover to subsets of the query terms. We define an *m-cover* for $m \leq n$ terms as an interval $[u, v]$ in the collection that contains an occurrence of at least m distinct terms in the query and does not contain a smaller interval that also contains m distinct terms. For simplicity we assume that terms are not repeated in q because the passage ranking method in this section does not take repeated query terms into account. For example, the set of 2-covers for the query vector ⟨"you", "quarrel", "sir"⟩ over the collection in Table 2.1 (page 50) consists of the intervals [2, 3], [3, 4], [4, 5], [5, 6], [8, 10], [10, 12], [12, 16], [24, 28]. Note that the interval [12, 24] is not included in the set because it contains [12, 16]. Note also that [24, 28] is included even though it cuts across the last three documents. In most applications m-covers such as this one would be filtered from the set before use. However, for simplicity we define m-covers without consideration for document boundaries and other structures and apply a postprocessing pass as appropriate.

The notion of an m-cover may be used to support snippet generation and arbitrary passage retrieval for question answering. Suppose we are looking for a snippet to display within an indexed Web page. Ideally we would display a snippet containing all the query terms in close proximity, but this goal is not always possible. Some of the query terms may appear only far apart, at the start and the end of the document. Some of the query terms may appear only in the title of the document and not in the body. Some of the query terms may not appear in the document at all. Instead, the snippet might be composed of text fragments corresponding to one or more m-covers that together contain as many of the terms as possible. Once an appropriate set of m-covers is determined, each could be extended to include the nearest sentence boundaries and then trimmed to fit into the required space.

In the context of question answering, a passage containing a strict subset of the query terms in close proximity may be more likely to yield an answer than a much longer passage containing all the query terms. For example, suppose we are answering the question "Who starred in the film *Shakespeare in Love*?" We might execute the query vector ⟨"starred", "film", "shakespeare", "love"⟩ and analyze the result for possible answers. A short 3-cover containing the terms "shakespeare" and "love" along with just one of the terms "starred" or "film" may be a better indicator of an answer than a longer passage containing all the terms, in which the answer may be mixed with other names and details. More generally, for applications such as snippet generation and question answering, we must select intervals by trading off length against the combination of query terms they contain.

9.6.1 Passage Scoring

Assume we have an interval $[u, v]$ that we wish to score with respect to a query $q = \langle t_1, t_2, ..., t_n \rangle$. Further assume that the interval contains a subset of the query terms $q' \subseteq q$, where $q' = \langle t_1', t_2', ..., t_m' \rangle$ and $m \leq n$. We assign a score to $[u, v]$, with length $l = v - u + 1$, by estimating the probability that a randomly selected interval of this length would contain at least one occurrence of these query terms. Like the method described in Section 9.5, the passage scoring method relates the actual distribution of terms to a random distribution.

For this purpose we model the collection as a sequence of independently generated terms and assume that there is a fixed probability p_t of a term $t \in q'$ matching at any given document location. Note that this assumption allows multiple terms to match at a particular location. Although unrealistic, this assumption is tolerable when p_t is small, as it is for most terms, and helps to simplify the derivation of a passage scoring formula.

Given the interval $[u, v]$ with length $l = v - u + 1$, the probability $p(t, l)$ that the interval contains one or more occurrences of t is

$$\begin{aligned} p(t,l) &= 1 - (1 - p_t)^l && (9.53) \\ &= 1 - (1 - lp_t + O({p_t}^2)) && (9.54) \\ &\approx l \cdot p_t \,. && (9.55) \end{aligned}$$

The probability that $[u, v]$ contains all the terms from q' is then

$$p(q', l) = \prod_{t \in q'} p(t, l) \approx \prod_{t \in q'} l \cdot p_t = l^m \cdot \prod_{t \in q'} p_t \,. \quad (9.56)$$

Finally, we estimate p_t as the collection frequency of the term t:

$$p_t = l_t / l_C \,, \quad (9.57)$$

where l_t is the total number of times t appears in the collection and l_C is the total length of the collection. Substituting and taking a negative logarithm (i.e., the self-information) gives

$$\sum_{t \in q'} (\log(l_C / l_t)) - m \cdot \log(l) \,. \quad (9.58)$$

The relationship between length and collection frequency in this equation resembles the relationship in Equation 9.32.

9.6.2 Implementation

Perhaps not surprisingly, the algorithm to locate m-covers is a simple extension of the adaptive algorithm appearing in Figure 2.10 (page 61). The details of the extended algorithm are given in Figure 9.2. For a given value of m the algorithm locates the next m-cover after a given position.

```
   nextCover (⟨t1, ..., tn⟩, position, m) ≡
1     for i ← 1 to n do
2        V[i] ← next(ti, position)
3     v ← mth largest element of V
4     if v = ∞ then
5        return [∞, ∞]
6     u ← v
7     for i ← 1 to n do
8        if V[i] < v and prev(ti, v + 1) < u then
9           u ← prev(ti, v + 1)
10    return [u, v]
```

Figure 9.2 Function to locate the next occurrence of an m-cover for the term vector $\langle t_1, ..., t_n \rangle$ after a given position. The integer array V is used to store intermediate calculations.

Lines 1–2 find the next occurrence of each individual term after this position. The m-th largest term becomes the end point (v) for the m-cover (line 3). For each term occurring before v we locate its last occurrence before v (lines 7–9). The smallest of these values becomes the start point (u) for the m-cover.

To generate all m-covers we repeatedly call this extended version of **nextCover** across the collection for all values of $m > 1$.

```
for m ← n down to 2 do
   u ← −∞
   while u < ∞ do
      [u, v] ← nextCover(⟨t1, t2, ..., tn⟩, u, m)
      if u ≠ ∞ then
         report the m-cover [u, v]
```

It is not necessary to explicitly generate 1-covers because these can be taken directly from the postings lists for the terms.

Depending on the application, it is not usually necessary to generate all m-covers across the entire collection. For snippet generation we are interested only in the m-covers contained in top ranking documents. For question answering we are interested only in a single best m-cover from a fixed number of documents. Clarke et al. (2006) discuss optimizations for fast m-cover generation in this second case.

9.7 Experimental Comparison

Table 9.1 Effectiveness measures for selected retrieval methods discussed in this chapter.

	TREC45				GOV2			
	1998		1999		2005		2006	
Method	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP
LMJM (Eq. 9.31)	0.390	0.179	0.432	0.209	0.416	0.211	0.494	0.257
LMD (Eq. 9.32)	0.450	0.193	0.428	0.226	0.484	0.244	0.580	0.293
DFR (Eq. 9.51/9.52)	0.426	0.183	0.446	0.216	0.465	0.248	0.550	0.269

Table 9.1 shows the effectiveness of the retrieval methods presented in this chapter. The numbers in this table may be compared with those in Tables 2.5 (page 72) and 8.2 (page 279). Results for the passage retrieval algorithm of Section 9.6 are not shown in the table because the method ranks passages, not documents. Although in principle it could be used for document ranking purposes, by assigning each document the score of its highest-scoring passage, this approach breaks down for one-word queries.

9.8 Further Reading

The seminal work on the language modeling approach to information retrieval, as presented in this chapter, includes that of Ponte and Croft (1998), Berger and Lafferty (1999), Zhai and Lafferty (2004), and the Ph.D. thesis of Hiemstra (2001). A volume edited by Croft and Lafferty consolidates much of the work up to 2003 (Croft and Lafferty, 2003). Lavrenko and Croft (2001) explore the role of relevance in the language modeling approach. Other early work includes that of Miller et al. (1999), who derive Equation 9.21 by assuming that queries are generated from documents using a hidden Markov model (HMM), where λ selects between a document state and a general language state, as represented by the collection. They then extend this HMM framework to incorporate pseudo-relevance feedback, proximity, and document priors.

Smoothing plays an important part in language modeling techniques across the full range of human language technologies. Chapter 6 of Manning and Schütze (1999) includes a general introduction to language modeling. Chen and Goodman (1998) provide a tutorial comparison of smoothing techniques.

The language modeling approach forms the foundation for a noticeable segment of current IR research. Most IR conference proceedings from the past several years contain papers that build upon or depend upon the insights of the language modeling approach. Although a complete

bibliography is outside the scope of this book, we provide a few examples of recent research: Metzler and Croft (2004) combine the language modeling approach with the probabilistic inference model (Turtle and Croft, 1991; Greiff et al., 1999); Cao et al. (2005) integrate term dependence into the language modeling approach; Tao and Zhai (2007) and Zhao and Yun (2009) integrate proximity measures into the language modeling approach; Cao et al. (2008) consider pseudo-relevance feedback within a language modeling framework. Lv and Zhai (2009) consider the integration of term positional information. Zhai (2008b) provides a recent survey of language modeling for information retrieval.

9.9 Exercises

Exercise 9.1 You have discovered that documents in a certain collection have a "half-life" of 30 days. After any 30-day period a document's prior probability of relevance $p(r \mid D)$ is half of what it was at the start of the period. Incorporate this information into Equation 9.8. Simplify the equation into a rank-equivalent form, making any assumptions you believe are reasonable.

Exercise 9.2 Show that models resulting from Dirichlet smoothing can be treated as probability distributions. That is, show $\sum_{t \in \mathcal{V}} \mathcal{M}_d^{\mu}(t) = 1$.

Exercise 9.3 (project exercise) Implement the LMD ranking formula (Equation 9.33). Test your implementation using the test collection developed in Exercise 2.13 or any other available collection, such as a TREC collection.

Exercise 9.4 (project exercise) Implement DFR ranking as described in Section 9.5. Test your implementation using the test collection developed in Exercise 2.13 or with any other available collection, such as a TREC collection.

Exercise 9.5 (project exercise) Implement the passage retrieval and scoring method described in Section 9.6. Use your implementation to provide result snippets for one of the document retrieval methods described in this book.

9.10 Bibliography

Amati, G., Carpineto, C., and Romano, G. (2003). Fondazione Ugo Bordoni at TREC 2003: Robust and Web Track. In *Proceedings of the 12th Text REtrieval Conference*. Gaithersburg, Maryland.

Amati, G., and van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. 20(4):357–389.

Berger, A., and Lafferty, J. (1999). Information retrieval as statistical translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229. Berkeley, California.

Cao, G., Nie, J.Y., and Bai, J. (2005). Integrating word relationships into language models. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–305. Salvador, Brazil.

Cao, G., Nie, J.Y., Gao, J., and Robertson, S. (2008). Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 243–250. Singapore.

Chen, S.F., and Goodman, J. (1998). *An Empirical Study of Smoothing Techniques for Language Modeling.* Technical Report TR-10-98. Aiken Computer Laboratory, Harvard University.

Clarke, C.L.A., Cormack, G.V., Lynam, T.R., and Terra, E.L. (2006). Question answering by passage selection. In Strzalkowski, T., and Harabagiu, S., editors, *Advances in Open Domain Question Answering.* Berlin, Germany: Springer.

Croft, W.B., and Lafferty, J., editors (2003). *Language Modeling for Information Retrieval.* Dordrecht, The Netherlands: Kluwer Academic Publishers.

Greiff, W.R., Croft, W.B., and Turtle, H. (1999). PIC matrices: A computationally tractable class of probabilistic query operators. *ACM Transactions on Information Systems*, 17(4):367–405.

Hiemstra, D. (2001). *Using language models for information retrieval.* Ph.D. thesis, University of Twente, The Netherlands.

Jelinek, F., and Mercer, R.L. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice.* Amsterdam, The Netherlands.

Lafferty, J., and Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119. New Orleans, Louisiana.

Lavrenko, V., and Croft, W.B. (2001). Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127. New Orleans, Louisiana.

Lv, Y., and Zhai, C. (2009). Positional language models for information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 299–306. Boston, Massachusetts.

Manning, C.D., and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing.* Cambridge, Massachusetts: MIT Press.

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

Miller, D. R. H., Leek, T., and Schwartz, R. M. (1999). A hidden Markov model information retrieval system. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214–221. Berkeley, California.

Plachouras, V., Ounis, I., Amati, G., and Rijsbergen, C. V. (2002). University of Glasgow at the Web Track of TREC 2002. In *Proceedings of the 11th Text REtrieval Conference*. Gaithersburg, Maryland.

Ponte, J. M., and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281. Melbourne, Australia.

Song, F., and Croft, W. B. (1999). A general language model for information retrieval. In *Proceedings of the 8th International Conference on Information and Knowledge Management*, pages 316–321. Kansas City, Missouri.

Tao, T., and Zhai, C. (2007). An exploration of proximity measures in information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 295–302. Amsterdam, The Netherlands.

Tellex, S., Katz, B., Lin, J., Fernandes, A., and Marton, G. (2003). Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. Toronto, Canada.

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

Zhai, C. (2008a). *Statistical Language Models for Information Retrieval*. Synthesis Lectures on Human Language Technologies: Morgan & Claypool.

Zhai, C. (2008b). Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval*, 2.

Zhai, C., and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342. New Orleans, Louisiana.

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

Zhao, J., and Yun, Y. (2009). A proximity language model for information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 291–298. Boston, Massachusetts.