The Information Retrieval Series

Donald Metzler

The Information Retrieval Series Volume 27

Donald Metzler

A Feature-Centric View of Information Retrieval

Springer

Donald Metzler
Natural Language Group
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
USA
metzler@isi.edu

ISSN 1387-5264 The Information Retrieval Series
ISBN 978-3-642-22897-1 e-ISBN 978-3-642-22898-8
DOI 10.1007/978-3-642-22898-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011937284

ACM Computing Classification (1998): H.3.3, G.3

Cover design: VTeX UAB, Lithuania

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To 晓黎

Acknowledgements

The material presented in this book represents the culmination of a line of research that began while I was a graduate student at the University of Massachusetts, Amherst. A great deal of the work presented here would not have been possible without a great cast of collaborators. First, and foremost, I must thank W. Bruce Croft for all of his support during my Ph.D. studies. I am also very grateful to Michael Bendersky, Jimmy Lin, Lidan Wang, and Hao Lang, all of whom made significant contributions that have greatly advanced and pushed this particular line of research well beyond where it was just a few years ago.

Contents

Chapter 1
Introduction

1 From Archie to Google and Beyond

Information retrieval is a dynamic discipline with a rich history. However, for much of its history, it had little or no impact on people's everyday lives. Many of the earliest consumers of information retrieval technologies were government researchers, scientists, and librarians. That began to change after the invention of the World Wide Web in the early 1990s.

Before the introduction of the Web, a number of information sources were available online. Most of the online information was published and controlled by government organizations or academic institutions. It was uncommon for everyday citizens to use these online systems, let alone publish their own content. The Web revolutionized the way that information was published. It allowed individuals and organizations to create content that was instantly available and easy to access. It also provided a way of linking content together, which was not possible with the older online systems. As computing costs decreased and online popularity increased, the amount of information available on the Web exploded.

As more electronic documents started appearing online, a natural desire to search the content arose. Various search tools were developed to help users find relevant files and documents. The earliest Internet search tools, Archie, Gopher, Veronica, and Jughead allowed users to search FTP servers. However, the popularity of FTP waned after the introduction of the Web. This ushered in a new era that gave rise to Web search engines. Unlike their predecessors, which were used by small fractions of the population, Web search engines such as Google are used every day by millions of users across the globe. Therefore, what started as a small, relatively unknown field of study, has evolved into an integral part of modern society.

2 The Academic and Industrial Perspectives

Yahoo and Google were both grown out of academic research projects. They currently are the two most popular commercial Web search engines in the United States.

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8_1,

Clearly, the academic research community, in the early days of the Web, was developing cutting edge search technologies. However, as the commercial search engines came of age, it become increasingly difficult for the academic researchers to keep up with the collection sizes and other critical research issues related to Web search. This caused a divide to form between the information retrieval research being done within academia and industry.

There are several reasons for this divide. First, as commercial search engines mature, they are able to collect more data in the form of query logs, click-through patterns, and other types of user data which are invaluable to Web search. The companies have little incentive to release these data to academia, especially amid growing privacy concerns. Second, commercial search engines have much more computing power than most academic research institutions. Therefore, they are able to crawl more Web pages, build larger indices, use real data streams, and experiment with much more costly computations. Finally, commercial search engines are very protective of their search algorithms and techniques and do not typically publish their findings in scholarly conferences and journals. This is not surprising, since revealing technical details of ranking functions may allow spammers and other malicious entities to adversely influence search results.

To put things into perspective, let us compare academic and industrial collection sizes. The Text REtrieval Conference (TREC), which was started in 1992, provides a set of standard, reusable test collections (i.e., document collection, queries, and relevance judgments) that most academic information retrieval researchers use when evaluating retrieval systems. One of the largest TREC test collections, called GOV2, is a 2004 crawl of the .gov top level domain. It consists of approximately 25 million Web pages (428 GB of text). In comparison, it is believed that Google has upwards of 25 billion items in its index, which is 1,000 times larger than GOV2. In an attempt to reduce the academic-industrial gap, in terms of collection sizes, the ClueWeb09 collection set was recently released, which consists of approximately 1 billion Web pages.

One of the goals of this work is to reduce the divide in understanding that exists between academic and commercial information retrieval systems with respect to large data sets. Many of the techniques and ideas developed here have been inspired by large test collections, such as GOV2 and ClueWeb09. While these collections are admittedly not Web-scale, they are significant and sizable improvements over the test collections that have been used to develop most of the current state-of-the-art information retrieval models.

3 Paradigm Shifts

As we just alluded to, large collections, such as those handled by commercial search engines, provide a new set of challenges for information retrieval researchers. In this work, we describe highly effective information retrieval models for both smaller, classical data sets, and larger Web collections. As we will show throughout this

work, the current state-of-the-art academic retrieval models are not robust enough to achieve consistently effective retrieval results on large collections.

Most of the these models are based on the so-called "bag of words" assumption. Under this assumption, text (e.g., queries and documents) are represented as unordered sets of terms. This means that any notion of term ordering is lost. For example, under this representation, the texts *the bear ate the human* and *the human ate the bear* are identical. However, these pieces of text clearly have different meanings. While this is an overly simplistic representation, very few have been able to develop non-bag of words retrieval models that are consistently and significantly better than the state-of-the-art bag of words models. Many researchers over the past few decades have tried in vain, but there has been very little success.

The ranking functions associated with bag of words retrieval models often consist of some combination of *term frequency* (TF) and *inverse document frequency* (IDF). The IDF component acts to discriminate between informative and non-informative query terms. Those terms that have a high IDF are considered more informative, because they rarely occur in the collection. On the other hand, terms that have a low IDF are considered uninformative, since they occur in many documents. As the number of documents in a collection increases, IDF becomes increasingly important in order to discriminate between those documents that only contain non-informative query terms and those that contain highly informative query terms.

On the other hand, the TF component, which is often normalized in some way with respect to the *document length*, is used to discriminate between documents that contain a query term several times and those that contain the term many times. This makes the assumption that documents that contain more mentions of a given query term are more "about" the given term and therefore are more likely to be relevant to the query. As we will discuss shortly, this is a bad assumption, especially as collection sizes increase and documents become noisier. The TF component becomes more important as documents get longer, since query terms are unlikely to occur more than one time in a very short document, and since long documents are more likely to contain more diverse term occurrence statistics.

Therefore, the TF and IDF components used within bag of words ranking functions, when combined together, discriminate along two dimensions—informativeness (IDF) and aboutness (TF). However, when dealing with large Web collections, a third dimension that we call noisiness enters the picture.

All collections, even small ones that consist entirely of news articles, contain some noise. However, large Web collections are likely to contain abundant amounts of noise. The standard TF and IDF features are not enough to overcome this noise. In fact, these features may actually help amplify the noise in some cases. Let us consider the query *habitat for humanity* run against a large Web collection. Using a state-of-the-art bag of words retrieval model, many of the top ranked results are relevant to the request. However, there are several results very high in the ranked list that do not contain a single occurrence of the term *humanity*. Instead, these documents contain hundreds of occurrences of the high IDF term *habitat*. These documents are ranked so highly because they contain many occurrences of a very high IDF term.

Documents that contain hundreds of occurrences of some high IDF term are going to result in poor, noisy matches for most bag of words models based on TF and IDF. Such documents may arise by coincidence, or a spammer who wishes to increase the ranking of a given Web page may "stuff" the page with such terms. In either case, it is very undesirable for these documents to be ranked highly.

Another more subtle way that noise may be introduced into bag of words matches happens when two or more query terms match a document, but the matches are random or unrelated to the query. For example, in the *habitat for humanity* case, consider a document that contains a paragraph that discusses habitat changes caused by global warming and another paragraph that discusses the negative impacts of global warming on humanity. Both the terms *habitat* and *humanity* will match this document, but the matches are unrelated to the query. That is, the terms just happened to match by chance. This is another example of noisy matches that can arise in large collections. In fact, as collection size grows, so does the chance that any two query terms will randomly match within some document.

Hence, new ranking function components, above and beyond TF and IDF must be used in order to reduce the number of noisy matches. There are a few ways to address this issue. First, one of the simplest ideas is to cap the TF component and not allow it to grow unbounded. While this addresses some noise issues, it fails to address the problem of randomly matching query terms. Second, in order to address the so-called term stuffing problem, anti-spam techniques may be developed in order to automatically detect malicious or misleading content. However, like capping TF, this only addresses some of the noise issues. Finally, term proximity features may be used in order to ensure that matches are not random and that they are somehow related to the query. For example, these types of features could be used to promote documents that contain the exact phrase *habitat for humanity* as opposed to those that simply contain random occurrences of the terms *habitat* and *humanity* on their own. It is this third option that we heavily explore within this work in order to overcome the limitations imposed by TF and IDF alone. It is important to notice that by using term position information, we are abandoning the bag of words assumption and move to a richer, more realistic text representation.

Aboutness, informativeness, and noisiness reflect the three primary information retrieval paradigm shifts. Here, a paradigm shift is a new way of approaching a problem with a given set of characteristics. The paradigm shifts are summarized in Fig. 1.1. The figure plots three data sets (CACM, TREC Disks 1 and 2, and GOV2) with respect to their average document length and the number of documents in the collection. As the figure shows, the TF paradigm shift moves along the average document length axis and the IDF shift moves along the number of documents axis. We also see that the noise shift moves along both axes, but is only present for large collections, such as GOV2. The newer ClueWeb09 collection, which is not shown in this plot, will likely require another (yet to be discovered) paradigm shift in retrieval methodologies to achieve maximal effectiveness.

We hypothesize that many of the previous attempts to go beyond the bag of words assumption have failed because of the small data sets used. In fact, most, if not all, of the previous research on non-bag of words model have been evaluated on test

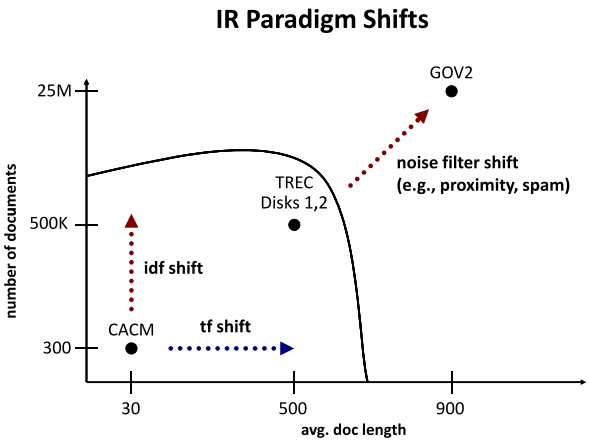

Fig. 1.1 A summary of the three primary information retrieval paradigm shifts. They include the TF shift (aboutness), the IDF shift (informativeness), and the noise shift (noisiness)

collections within the region shown in Fig. 1.1. Poor, or inconclusive results, were achieved because the data sets did not exhibit the characteristics necessary to exploit the noise reducing features associated with non-bag of words models. Therefore, new models that go beyond the bag of words assumption should be tested on large, noisy data sets in order to properly evaluate their full potential.

4 A Robust Retrieval Model

In this work, we describe a robust statistical information retrieval model based on Markov random fields. In particular, the model is designed to support the following desiderata:

1. Support basic information retrieval tasks (e.g., ranking, query expansion, etc.).
2. Easily and intuitively model query term dependencies.
3. Handle arbitrary textual and non-textual features.
4. Consistently and significantly improve effectiveness over bag of words models across a wide range of tasks and data sets.

The model we describe goes beyond the bag of words assumption in two ways. First, the model can easily exploit various types of dependencies that exist between query terms. This eliminates the term independence assumption that often accompanies bag of words models. Second, arbitrary textual or non-textual features can be used within the model. Thus, it is possible to use simple features such as TF and IDF, or more complex features, such as those based on term proximity. Other possible features include PageRank, inlink count, readability, spam probability, among others. None of the current state-of-the-art models allow arbitrary features to be incorporated as easily as the Markov random field model.

As we will show, combining term dependencies and arbitrary features results in a very robust, powerful retrieval model. Within the model, we describe several extensions, such as an automatic feature selection algorithm and a query expansion

framework. The resulting model and extensions provide a flexible framework for highly effective retrieval across a wide range of tasks and data sets.

5 Outline

The remainder of this work is laid out as follows.

- Chapter 2 summarizes several important classical retrieval models. The chapter divides the discussion of the models into non-bag of words models (e.g., Binary Independence Model, BM25, language modeling, etc.) and models that go beyond the bag of words assumption (n-gram language models, Inference Network Model, etc.). It concludes with a brief discussion of the current state-of-the-art.
- Chapter 3 motivates and covers the basics of the Markov random field model for information retrieval, which serves as the basis for exploring feature-based retrieval models throughout the remainder of the work. The chapter begins with a theoretical treatment of the model and concludes with a detailed analysis of the practical aspects of the model.
- Chapter 4 explains how the Markov random field model can be used for robust, highly effective feature-based query expansion via the use of a method called Latent Concept Expansion (LCE). LCE and several of its extensions are described in detail.
- Chapter 5 describes a powerful extension of the basic Markov random field model that supports query-dependent term weighting. Unlike most existing retrieval models that weight all features the same, regardless of the query, the approach described in this chapter provides a means for adaptively weighting the importance of each feature based on properties of the query.
- Chapter 6 covers a number of techniques that can be used to estimate the parameters of feature-based models. The emphasis of this chapter is on simple techniques that can be used to learn the parameters of models typically encountered and used within research environments. More sophisticated approaches that are more well-suited for estimating parameters within industrial settings are also covered.

In addition to the primary technical content, this work also includes the two following appendices that are meant to provide readers with additional background information.

- Appendix A covers the anatomy of TREC test collections and summarizes the data sets used throughout this work.
- Appendix B explains how the various data sets used throughout this work are computed.

Chapter 2
Classical Retrieval Models

1 Overview

This chapter provides a survey of classical information retrieval models. There is no standard or formal way of describing retrieval models. However, most models can be uniquely described in terms of their *document representation*, *query representation*, and a *ranking function*. In the most common scenario, the ranking function takes a document and query representation as input, and outputs a score that is indicative of how relevant the document is to the query.

Throughout our discussion we will refer to documents, queries, and relevance. However, it should be noted that these terms are actually intended to be used quite generally. For example, in an automatic question answering system, a 'document' is an answer, a 'query' is a question, and 'relevance' is defined according to whether or not the answer, with regard to the question, is correct (Voorhees 1999). Other examples include image retrieval, where the 'documents' are images, and a query-query similarity task, where the 'documents' are queries (Metzler et al. 2007; Metzler and Manmatha 2004).

2 Bag of Words Models

We begin by looking at so-called bag of words retrieval models. As we will see, these models make use of many different types of document representations, query representations, and ranking functions. However, the one thing that all of these models have in common is the fact that term order is ignored when constructing the document and query representations. That is, given a document A and A^{π}, a permutation of A, the representations of A and A^{π} are identical.

This assumption is obviously overly simple. It conflates many texts that have very different semantic meanings into a single form. For example, the texts *the human ate the bear* and *the bear ate the human* have very different semantic meanings, but are represented the same way under the bag of words assumption. While it is not known

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8_2,

how the human brain represents text, it is unlikely that term ordering is completely ignored. Indeed, term orderings play an important role in semantics. Therefore, it is difficult to expect a computer, using such a simple representation, to determine relevance as accurately as a human assessor can.

Despite the fact that the bag of words assumption is so simple, some believe that it is a reasonable first approximation (Lavrenko 2004; Salton and Buckley 1988). For example, in 1988, Salton and Buckley stated:

> In reviewing the extensive literature accumulated during the past 25 years in the area of retrieval system evaluation, the overwhelming evidence is that the judicious use of single term identifiers is preferable to the incorporation of more complex entities...

Although this was written in 1988, there has been little, if any, conclusive evidence to discredit this claim.

Throughout this work, we argue, and validate experimentally, that models based on the bag of words assumption are inferior to models that consider richer representations. We acknowledge that single term identifiers are likely to be the most powerful and discriminative types of identifiers. However, we describe a number of other identifiers (features) that can be used to significantly improve retrieval effectiveness well beyond the state-of-the-art bag of words models discussed in this chapter.

Finally, one important thing to note is that the bag of words assumption does not force the underlying model to treat term occurrences as independent events. For example, it is possible for bag of words models to incorporate term co-occurrence statistics into the ranking function, thereby modeling one very basic form of term dependence (Berger and Lafferty 1999; van Rijsbergen 1977; Wei and Croft 2007). We return to the subject of term dependence models shortly.

2.1 Binary Independence Retrieval Model

We begin our discussion of bag of words models by describing the Binary Independence Retrieval (BIR) model, which is one of the earliest probabilistic models for information retrieval (Robertson and Spärck Jones 1976). The model is also known by many other names including the Okapi model, the City model, the Robertson–Sparck Jones model, and the classical probabilistic model.

Under the BIR model, documents are represented as binary vectors d indexed over a fixed vocabulary $\mathcal{V}$ (i.e., $d \in \{0, 1\}^{|\mathcal{V}|}$). The vocabulary typically consists of single terms, but may also include more general concepts, such as phrases. If a term, $w \in \mathcal{V}$, occurs one or more times in a document, then $d_w = 1$, otherwise $d_w = 0$. Documents are then ranked according to the *Probability Ranking Principle* (PRP) (Robertson 1977).

> **Probability Ranking Principle.** 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.

The PRP states that documents should be ranked in decreasing order of probability of relevance given all of the available evidence. Thus, documents are ranked according to $P(R = 1|d)$, where R is a binary random variable that models relevance. Also, in order to make parameter estimation feasible, the model assumes that the term occurrence variables (d_w) are conditionally independent, given a relevance class. Using these assumptions, we now derive the BIR model's ranking function:

$$\begin{aligned} P(R=1|d) &\overset{\text{rank}}{=} \frac{P(R=1|d)}{P(R=0|d)} \\ &= \frac{P(d|R=1)P(R=1)}{P(d|R=0)P(R=0)} \\ &\overset{\text{rank}}{=} \frac{P(d|R=1)}{P(d|R=0)} \\ &= \prod_{w\in\mathcal{V}} \frac{P(d_w=1|R=1)^{\delta_w} P(d_w=0|R=1)^{1-\delta_w}}{P(d_w=1|R=0)^{\delta_w} P(d_w=0|R=0)^{1-\delta_w}} \\ &\overset{\text{rank}}{=} \sum_{w\,:\,\delta_w=1} \log \frac{P(d_w=1|R=1)P(d_w=0|R=0)}{P(d_w=0|R=1)P(d_w=1|R=0)}, \end{aligned} \tag{2.1}$$

where $\overset{\text{rank}}{=}$ denotes rank equivalence[1] and δ_w is 1 if term w occurs in the document and 0 otherwise.

Although the model, at first glance, appears to make the relatively strong assumption of conditional independence, the model is actually much less restrictive (Cooper 1991). Instead, as Cooper shows, the model allows dependencies to exist between terms, but requires the strength of the dependence to be equal in both the relevant ($R = 1$) and the non-relevant ($R = 0$) sets of documents, thereby linking the dependence. Therefore, instead of assuming conditional independence, the model actually assumes *linked dependence*. Even though this is a weaker assumption, it is still restrictive and unlikely to hold, in general.

One criticism of the model is that there is no explicit query involved. Rather, the model assumes that $P(R = 1|d)$ is implicitly conditioned on some underlying information need. This simplifies the model to a certain extent, but raises certain theoretical issues when researchers try to incorporate aspects of the query into the model.

In order to use the model for ranking, $P(d_w = 1|R = 0)$ and $P(d_w = 1|R = 1)$ must be estimated for all w. This is a total of $2|\mathcal{V}|$ parameters, which for large vocabularies, is an immense number of parameters. We now describe how these parameters are typically estimated when we have relevance information and when we do not.

[1] Two functions are defined to be rank equivalent if they guaranteed to produce the same ordering of items when sorted according to function value. Monotonic transforms, scaling (by a constant), and translation (by a constant) are all examples of rank-preserving operations.

2.1.1 Estimation with Relevance Information

If we are given relevance information for a given information need then estimation is straightforward. Assume we are given a set of documents that are known to be relevant to the information need, as well as a set that are known to be non-relevant. In the simplest case, we can get this information from a user who has manually judged a set of documents. Using this information, we obtain the following estimates:

$$P(d_w = 1|R = 0) = \frac{nr_w + \alpha_{nr}}{NR + \alpha_{nr} + \beta_{nr}}, \tag{2.2}$$

$$P(d_w = 1|R = 1) = \frac{r_w + \alpha_r}{R + \alpha_r + \beta_r}, \tag{2.3}$$

where nr_w is the number of judged non-relevant documents that term w occurs in, NR is the total number of judged non-relevant documents, r_w is the number of judged relevant documents term w occurs in, R is the total number of documents judged relevant, and α and β are smoothing parameters that avoid zero probabilities and help overcome data sparsity. Historically, the smoothing parameters have been set to $\alpha_{nr} = \alpha_r = 0.5$ and $\beta_{nr} = \beta_r = 0$, although there is no reason to believe these are the optimal settings.

2.1.2 Estimation Without Relevance Information

As we just showed, estimation with relevance information is rather straightforward. However, for most queries, a system will not have access to relevance information. Not surprisingly, parameter estimation under this scenario proves to be much more challenging. In order to use the model when there is no relevance information, a number of assumptions must be made (Croft and Harper 1979; Robertson and Walker 1997). These assumptions include:

1. $P(d_w = 1|R = 0) = P(d_w = 1|R = 1)$ for all terms that do not occur in the query.
2. $P(d_w = 1|R = 1) = 0.5$ for all terms that occur in the query.
3. $P(d_w = 1|R = 0) = \frac{df_w + \alpha_{nr}}{N + \alpha_{nr} + \beta_{nr}}$ for all terms that occur in the query, where df_w is the number of documents that w occurs in and N is the number of documents in the collection.

The first two assumptions are poor and very unlikely to hold true. However, they greatly simplify how the ranking function is computed, which is necessary, given the lack of relevance information.

The third assumption is the most reasonable. It assumes that we have observed no relevant documents and that every document in the collection is non-relevant. This information is treated as real relevance information and plugged into Eq. 2.2 to derive the estimate of $P(d_w = 1|R = 0)$ shown above. Of course, this assumption is not completely accurate. However, it has been shown that the collection, as a whole, acts as a good proxy for modeling the set of non-relevant documents.

After invoking these assumptions, the BIR model ranking function simplifies to the following form:

$$P(R=1|d) \stackrel{\text{rank}}{=} \sum_{w\,:\,\delta_w=1 \wedge w \in Q} \log \frac{N - df_w + 0.5}{df_w + 0.5}, \tag{2.4}$$

where $w \in Q$ indicates that term w occurs in the query and we set $\alpha_{nr} = \beta_{nr} = 0.5$, which is commonly used in this scenario. The term inside of the summation is the well-known Okapi IDF. This derivation provides one of the many theoretical justifications for the importance of IDF (Robertson 2004).

2.1.3 Tree Dependence Model

The tree dependence model is one of many extensions that have been proposed for the BIR model (van Rijsbergen 1977). It attempts to model first-order dependencies between terms by making use of the Chow expansion (Chow and Liu 1968). The Chow expansion is a method for approximating a joint distribution in terms of first-order dependencies.

The tree dependence model constructs a weighted, undirected graph G for both the relevant and non-relevant sets, such that the vertices of G are the terms in the vocabulary, there exists an edge between every pair of terms, and the edge weights are the expected mutual information between the two terms, which is computed as

$$I(t_1, t_2|R) = \sum_{\delta_1 \in \{0,1\}} \sum_{\delta_2 \in \{0,1\}} P(\delta_1, \delta_2|R) \log \frac{P(\delta_1, \delta_2|R)}{P(\delta_1|R)P(\delta_2|R)}, \tag{2.5}$$

where t_1 and t_2 are terms, and $P(d_t|R)$ and $P(d_{t_1}, d_{t_2}|R)$ are typically estimated using relevance information.

This graph is then used in the following way. First, a maximum spanning tree is constructed from the graph. Next, an arbitrary node is chosen as the root, which allows all of the edges to be directionalized. Finally, the directed graph is used to compute a first-order approximation to the joint distribution over all of the terms by assuming that a term is dependent on its parent term[2].

Figure 2.1 show an example maximum spanning tree over a graph with five terms. As we see, the root node is term D. The first-order approximation of the joint distribution can then be written as

$$P(A, B, C, D, E) = P(D)P(A|D)P(C|D)P(B|C)P(E|C). \tag{2.6}$$

In the context of the BIR model, this approximation can be used to compute $P(d|R=0)$ and $P(d|R=1)$, which results in the following ranking function:

$$\sum_{w\,:\,\delta_w=1} \log \frac{P(d_w=1|d_{\pi_w}=\delta_{\pi_w}, R=1)P(d_w=0|d_{\pi_w}=\delta_{\pi_w}, R=0)}{P(d_w=0|d_{\pi_w}=\delta_{\pi_w}, R=1)P(d_w=1|d_{\pi_w}=\delta_{\pi_w}, R=0)}, \tag{2.7}$$

where π_w is the parent term of w in the spanning tree.

[2]The root node, which has no parents, is assumed to not be dependent on any other terms.

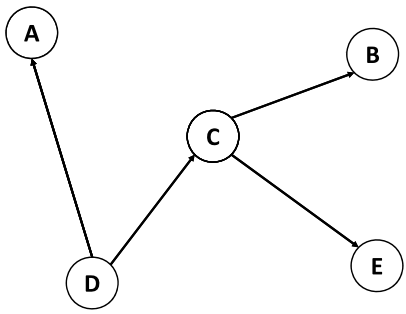

Fig. 2.1 An example spanning tree for five terms, rooted at D

Parameter estimation in the tree dependence model is even more difficult than in the BIR model because the data sparseness problem is only exacerbated by the large number of conditionals that must be estimated. Furthermore, the model has never shown consistent improvements in effectiveness over the BIR model (van Rijsbergen 1977). Therefore, the model is interesting mostly from a theoretical and historical perspective.

2.2 2-Poisson Model

As we just explained, the BIR model represents documents as binary vectors. This representation can only capture whether or not a term occurs in a document. It ignores the number of times the term occurs (term frequency), which, as we discussed in Chap. 1, is important, especially as document lengths increase.

The 2-Poisson model was proposed to overcome this limitation (Harter 1975; Robertson et al. 1980). Under this model, documents are represented as vectors of term frequencies. The vector is indexed over a fixed vocabulary $\mathcal{V}$, and thus $d \in \mathbb{N}^{|\mathcal{V}|}$ for every document d.

The 2-Poisson ranking function is derived analogously to the BIR ranking function, as follows:

$$\begin{aligned} P(R=1|d) &\overset{\text{rank}}{=} \frac{P(d|R=1)}{P(d|R=0)} \\ &= \prod_{w \in \mathcal{V}} \frac{P(d_w = tf_w|R=1)}{P(d_w = tf_w|R=0)} \\ &\overset{\text{rank}}{=} \sum_{w\,:\,tf_w>0} \log \frac{P(d_w = tf_w|R=1)P(d_w=0|R=0)}{P(d_w=0|R=1)P(d_w = tf_w|R=0)}, \end{aligned} \tag{2.8}$$

where term frequencies, as before, as assumed to be conditionally dependent[3] and tf_w denotes the number of times that term w occurs in the document.

[3] As in the BIR model, the weaker linked dependence assumption holds.

To use the model, $P(d_w = tf_w|R = 0)$ and $P(d_w = tf_w|R = 1)$ must be estimated. As its name implies, the 2-Poisson model assumes that term frequencies are distributed according to a mixture of two Poissons, as given by

$$P(d_w = tf_w|R = 1) = P(E = 1|R = 1)\frac{e^{-\alpha_w}\alpha_w^{tf_w}}{tf_w!} + P(E = 0|R = 1)\frac{e^{-\beta_w}\beta_w^{tf_w}}{tf_w!}, \tag{2.9}$$

$$P(d_w = tf_w|R = 0) = P(E = 1|R = 0)\frac{e^{-\alpha_w}\alpha_w^{tf_w}}{tf_w!} + P(E = 0|R = 0)\frac{e^{-\beta_w}\beta_w^{tf_w}}{tf_w!}, \tag{2.10}$$

where α_w and β_w are parameters of the Poisson distributions, and E is a binary variable that represents *eliteness* (Harter 1975; Robertson and Walker 1994).

Eliteness is a hidden, or latent, variable that reflects whether or not a given term is actually "about" a document or not. Therefore, elite terms are "about" a document, and non-elite terms are not. Given the subjective nature of its definition, it is very difficult to quantify or actually model eliteness in practice. For this reason, the model is interesting purely from a theoretical point of view.

2.3 BM25 Model

The BM25 model, proposed by Robertson and Walker, is an empirical, hand-crafted approximation of the 2-Poisson model (Robertson and Walker 1994). It was first introduced at TREC in 1995, and has been widely used ever since (Robertson et al. 1994). The ranking function is given by

$$P(R = 1|d) \approx \sum_{w \in Q \cap d} tf_{w,Q}\frac{(k_1 + 1)tf_{w,d}}{k_1((1 - b) + b\frac{|d|}{|d|_{\text{avg}}}) + tf_{w,d}} \log\frac{N - df_w + 0.5}{df_w + 0.5}, \tag{2.11}$$

where $tf_{w,Q}$ is the number of times term w occurs in the query, $tf_{w,d}$ is the number of times term w occurs in the document, $|d|$ is the number of terms in the document, $|d|_{\text{avg}}$ is the average document length, N is the number of documents in the collection, df_w is the number of documents term w occurs in the collection, and k_1 and b are model parameters.

The model is very simple to implement and, with carefully chosen model parameters, has been shown to consistently achieve state-of-the-art effectiveness. Unfortunately, the model, despite being inspired by the 2-Poisson model, is heuristic and has no built-in mechanism for modeling term dependencies. It has been shown that it is possible to incorporate query independent features, such as PageRank into the model (Craswell et al. 2005b), as well as term proximity information (Büttcher et

al. 2006a). However, these improvements are heuristic and are unrelated to the assumed underlying 2-Poisson model. Furthermore, there is no convenient, formally motivated framework for easily adding other types of features to the model, which limits the usefulness of the model.

2.4 Unigram Language Models

Language modeling, a statistical technique first applied to speech recognition (Rosenfeld 2000), has also been successfully applied to information retrieval (Ponte and Croft 1998). Since its introduction, it has grown in popularity and has been proven to be a robust, highly effective retrieval model. In the context of information retrieval, language models are statistical models of text generation. In this section we will describe the simplest type of language model, the unigram model, which is based on the bag of words assumption. Later, we describe more complex language models.

The most common strategy for using language models for information retrieval is called the *query likelihood* approach. Given a query Q, documents are ranked according to the likelihood that the query was generated, given document D as evidence. Typically, for the sake of smoothing the document model, a Bayesian estimate is used. Using the approach, documents are ranked according to

$$\begin{aligned} P(Q|D) &= \prod_{q \in Q} P(q|D) \\ &= \prod_{q \in Q} \int_{\theta_D} P(q|\theta_D) P(\theta_D|D) \\ &\propto \prod_{q \in Q} \int_{\theta_D} P(q|\theta_D) P(D|\theta_D) P(\theta_D), \end{aligned} \tag{2.12}$$

where the unigram language model (θ_D) is typically a multinomial distribution over a fixed vocabulary (Song and Croft 1999). In addition, for computational simplicity, it is assumed that $P(\theta_D)$ is Dirichlet. This is typically called Bayesian or Dirichlet smoothing (Zhai and Lafferty 2001b). Under these assumptions, we obtain the following estimate for $P(w|D)$:

$$P(w|D) = \frac{tf_{w,D} + \mu P(w|C)}{|D| + \mu}, \tag{2.13}$$

where it is assumed that the Dirichlet parameters are $\alpha_w = \mu P(w|C)$, where μ is a model hyperparameter and $P(w|C) = \frac{cf_w}{|C|}$, with cf_w being the number of times term w occurs in the collection and $|C|$ is the total number of terms in the collection. Using this estimate, documents are then ranked according to:

$$P(Q|D) \stackrel{\text{rank}}{=} \sum_{q \in Q} \log \frac{tf_{w,D} + \mu P(w|C)}{|D| + \mu}$$

$$\stackrel{\text{rank}}{=} \sum_{q \in Q \cap D} \log \frac{\frac{tf_{w,D}+\mu P(w|C)}{|D|+\mu}}{\frac{\mu P(w|C)}{|D|+\mu}} + \sum_{q \in Q} \log \frac{\mu P(w|C)}{|D|+\mu}$$

$$\stackrel{\text{rank}}{=} \sum_{q \in Q \cap D} \log\left[1 + \frac{tf_{w,D}}{\mu} \cdot \frac{|C|}{cf_w}\right] - |Q| \log\big(|D|+\mu\big) \qquad (2.14)$$

which can be interpreted as another variant on the standard *tf.idf* formula with built-in document length normalization.

Although we described Bayesian smoothing here, it should be noted that many other types of smoothing are possible (Zhai and Lafferty 2001b, 2002, 2004). Also, distributions other than the multinomial have been proposed for modeling documents and queries, such as the multiple-Bernoulli distribution (Metzler et al. 2004a).

Even though language modeling is more formally motivated than BM25, the ranking functions are quite similar, with both relying on the standard *tf*, *idf*, and document length normalization components. Many of the problems with BM25 are also carried over to this model. For example, language models are models of *text* generation and therefore it is difficult to incorporate non-textual features into the model. Arbitrary query independent features are often encoded using a document prior (Kraaij et al. 2002), but this is not applicable to query dependent features. One possible solution is to use a general Naïve Bayes model, but such a model, by its very nature, is incapable of modeling term dependencies. As we will soon show, there have been a number of models proposed to address this problem using more complex language models.

2.5 Other Bag of Words Models

The axiomatic approach to retrieval (Fang and Zhai 2005) and the divergence from randomness model (Amati and van Rijsbergen 2002) are two recently proposed bag of words retrieval models. The ranking functions of these two models are variants on the *tf.idf* theme. While the models shed interesting insights into retrieval modeling, they have not been shown to be significantly better or more flexible than language modeling or BM25.

The tree dependence model was one of the first bag of words models that attempted to capture the dependence that exists between terms. Several other bag of words models have been proposed to capture dependencies, as well. Early examples include the Bahadur Lazarsfeld expansion (BLE) (Losee 1994) which is an exact, but computationally intensive, method of modeling high order dependencies, and the Generalized Dependence Model, which generalizes both the tree dependence model and the BLE expansion (Yu et al. 1983).

Other examples include latent semantic analysis (Deerwester et al. 1990; Hofmann 1999), term association models (Berger and Lafferty 1999; Fang and Zhai 2006; Spärck Jones 1971; Wei and Croft 2007), cluster-based language models (Diaz 2005; Kurland and Lee 2004; Liu and Croft 2004), topic models (Blei et al.

2003a, 2003b; Griffiths et al. 2005; Wei and Croft 2006), and pseudo-relevance feedback (Diaz and Metzler 2006; Lavrenko and Croft 2001; Zhai and Lafferty 2001a).

Despite the increased complexity, many of these have failed to yield substantial improvements in effectiveness. Several of the models, including cluster-based language modeling and some of the topic models, have shown significant improvements in retrieval effectiveness. However, these models are often computationally intensive, making them impractical to apply to Web-scale collections. The model that we focus on in this work does not require any expensive computations and can easily be applied to very large collections. In addition, it allows other types of term dependence features, beyond simple co-occurrence statistics, to be used, thus making it more robust and practical for a wide range of tasks and data sets.

3 Models That Go Beyond the Bag of Words Assumption

We now describe a set of models that go beyond the bag of words assumption. These models are typically more complex, less efficient, and less effective. For these reasons they are not widely used within the information retrieval community. However, it is important to describe the breadth of work done in this area in order to provide a clear picture of the difficulty involved with developing highly effective models that go beyond the bag of words assumption.

3.1 n-Gram Language Models

In this section we describe n-gram language models (for $n > 1$). These models are simple generalizations of the unigram language model approach that take context into account. That is, n-gram language models generate terms by conditioning on the previous $n - 1$ terms encountered. In a unigram model, generating the term *Lincoln* is equally likely regardless of the previous term. In a bigram model ($n = 2$), *Lincoln* has a higher likelihood of being generated after *president* than after *brick*, for example. Therefore, n-gram models capture the sequential structure of language generation.

As with unigram language models, documents are ranked according to query likelihood, which is computed as

$$P(Q|D) = \prod_{i=1}^{|Q|} P(q_i|q_{i-1}, \ldots, q_{i-n+1}, D), \tag{2.15}$$

where $P(q_i|q_{i-1}, \ldots, q_{i-n+1}, D)$ can be estimated in a number of ways (Gao et al. 2004; Song and Croft 1999; Srikanth and Srihari 2002), many of which include some form of backoff to a unigram model. One way of estimating bigram probabilities, from Gao et al. (2004), is

$$P(w_i|w_{i-1}, D) = (1-\lambda_1)\left[(1-\lambda_2)\frac{tf_{w_i,w_{i-1},D}}{tf_{w_{i-1},D}} + \lambda_2 \frac{tf_{w_i,D}}{|D|}\right] + \lambda_1\left[(1-\lambda_3)\frac{cf_{w_i,w_{i-1}}}{cf_{w_{i-1}}} + \lambda_3 \frac{cf_{w_i}}{|C|}\right], \quad (2.16)$$

where λ_1, λ_2, and λ_3 are free parameters that control the smoothing.

Gao et al. showed that this model consistently outperformed unigram language models across a number of data sets using description-length queries (Gao et al. 2004). Unfortunately, the model, as described, performs poorly on title-length queries. The model is a generalization of Jelinek–Mercer smoothing, which is known to work well on longer queries (Zhai and Lafferty 2004). Therefore, the model must be adjusted to be more like Dirichlet smoothing in order to perform well on title queries. This can be achieved by setting $\lambda_1 = \frac{\mu_1}{\mu_1+|D|}$. We note that this modification does not follow naturally or formally from some underlying model. Instead it is a heuristic modification that only works because it makes the ranking function more like the Dirichlet ranking function.

Although n-gram language models capture the relationship between terms better, they are still not adequately robust for our needs. One criticism of such models is that they rely on evidence from the previous $n-1$ terms, when in fact the next $n-1$ terms might provide just as strong evidence. Consider the text *white house rose garden*. In a reasonable bigram language model of general English, $P(house|white)$ would be assigned a high probability, but $P(rose|house)$ would not. Therefore, under the bigram model, the likelihood of this sequence may actually be underestimated. However, conditioning on both past and future words could overcome such a problem. It is also noted that n-grams are typically overly strict, in that they do not allow the modeling of longer-range, unordered dependencies, such as the fact that two terms tend to often appear, not necessarily in order, within close proximity to each other. As we will show, the Markov random field model is capable of handling a wide range of dependencies, including those that n-gram language models are not capable of.

3.2 Indri Inference Network Model

The retrieval model implemented in the Indri search engine (Strohman et al. 2004) is an enhanced version of the model described in Metzler and Croft (2004), which combines the language modeling (Song and Croft 1999) and inference network (Turtle and Croft 1991) approaches to information retrieval. The resulting model allows rich, structured queries to be evaluated using language modeling estimates within the network. Figure 2.2 shows a graphical model representation of the network. Within the model, documents are ranked according to $P(I|D, \alpha, \beta)$, the belief the information need I is met given document D and hyperparameters α and β as evidence.

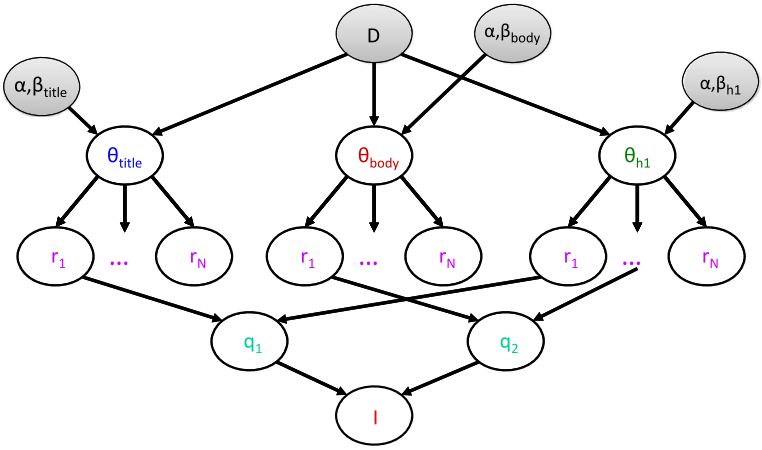

Fig. 2.2 Indri's inference network retrieval model

$$f\left(\texttt{A B C A B},\begin{bmatrix} A \\ B \\ C \\ AA \\ AB \\ AC \\ BA \\ BB \\ BC \\ CA \\ CB \\ CC \end{bmatrix}\right) = \left\{ \begin{bmatrix} 1\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 0\\1\\0\\0\\0\\0\\0\\0\\1\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 0\\0\\1\\0\\0\\0\\0\\0\\0\\1\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0 \end{bmatrix} \right\}$$

Fig. 2.3 Example Indri document representation for the document `A B C A B`. The features correspond to the single terms *A*, *B*, *C*, and bigrams "*AA*", "*AB*", "*AC*", "*BA*", "*BB*", "*BC*", "*CA*", "*CB*", and "*CC*", respectively. The function f takes a document and set of features as input and outputs a document representation

3.2.1 Document Representation

Typically, in the language modeling framework, documents are represented as a sequence of terms. Based on this sequence, a multinomial language model over the vocabulary is estimated. However, it is often the case that we wish to model more interesting text phenomenon, such as phrases or the *absence* of a term. Therefore, a different representation scheme is necessary. In the Indri model, documents are represented as multisets of binary feature vectors. Given a document, a feature vector is extracted for *every* position within the document. Figure 2.3 provides an example representation using this scheme. As the figure shows, the document is represented by a set of five vectors, one for each position within the document. This repre-

sentation is very general and provides a way of modeling almost arbitrary textual features.

This representation moves away from modeling text toward modeling features of text. Throughout the remainder of this section we refer to such models as language models, although they really are better described as *language feature models*.

3.2.2 Language Models

Since the document event space is binary, it is no longer appropriate to estimate multinomial language models for each document. Instead, multiple-Bernoulli models are estimated, as in Model B of Metzler et al. (2004a). This overcomes the theoretical issues encountered in Metzler and Croft (2004). Note that the multiple-Bernoulli model imposes the assumption that the features (r_i's) are independent, which of course may be a poor assumption depending on the feature set.

A Bayesian approach is taken and a multiple-Beta prior is imposed over the distribution of language models (θ). The Beta is chosen for simplicity, as it is the conjugate prior to the Bernoulli distribution. Thus, $P(D|\theta)$ is distributed according to *Multi-Bernoulli*(θ) and $P(\theta|\alpha, \beta)$ is distributed according to *Multi-Beta*(α, β). Hence, the belief at node θ is computed as

$$\begin{aligned} P(\theta_i|D, \alpha, \beta) &= \frac{P(D|\theta_i)P(\theta_i|\alpha_i, \beta_i)}{\int_{\theta_i} P(D|\theta_i)P(\theta_i|\alpha_i, \beta_i)} \\ &= Beta\big(\#(r_i, D) + \alpha_i, |D| - \#(r_i, D) + \beta_i\big) \end{aligned} \tag{2.17}$$

for each i where $\#(r_i, D)$ is the number of times feature r_i is set to 1 in document D's multiset of feature vectors.

Such a model is estimated using the entire document text. Additionally, if a document is structured, as HTML, SGML, and XML documents are, then language models can be estimated for each field. To do so, we treat all of the text in that appears within a given field as a pseudo-document. For example, a model can be estimated for all of the text that appears within the **h1** tags of a Web page document.

3.2.3 Representation Nodes

The r_i nodes correspond to document features that can be represented in an Indri structured query. There is a one-to-one correspondence between r_i nodes and the features used to represent the document. Therefore, the r_i nodes represent binary events that feature i is observed.

Indri implements a number of textual features, including single terms, #N (ordered window N), and #uwN (unordered window N). Please refer to Metzler and Croft (2004) for more details on these operators.

Using the framework developed thus far, the belief at a given representation node is computed as

$$P(r_i = 1|D, \alpha, \beta) = \int_{\theta_i} P(r_i = 1|\theta_i) P(\theta_i|D, \alpha_i, \beta_i)$$
$$= E\big[P(r_i|\theta_i)\big]$$
$$= \frac{\#(r_i, D) + \alpha_i}{|D| + \alpha_i + \beta_i}. \tag{2.18}$$

Furthermore, selecting $\alpha_i = \mu P(r_i = 1|C)$ and $\beta_i = \mu(1 - P(r_i = 1|C))$ results in the multiple-Bernoulli model equivalent of the multinomial model's Dirichlet smoothing (Zhai and Lafferty 2001b) estimate:

$$P(r_i|D, \alpha, \beta) = \frac{\#(r_i, D) + \mu P(r_i|C)}{|D| + \mu}, \tag{2.19}$$

where μ acts as a tunable smoothing parameter.

3.2.4 Query Nodes

The query node operators are soft probabilistic operators that are used to combine evidence within the network. The operators are primarily used to combine evidence from representation nodes and other query nodes. The operators implemented in Indri are #combine (same as #and), #weight (same as #wand), #or, #not, #sum, #wsum, and #max (Metzler and Croft 2004).

The information need node, I, is also a query node that acts to combine all of the evidence of the query into a single belief. It is this belief (i.e., $P(I = 1|D, \alpha, \beta)$) that is used to rank documents. Therefore, the ranking function is defined in terms of query and representation nodes. For example, consider the following Indri query:

```
#weight(1.0 #or(american #1(united states)) 2.0
                presidents)
```

For this query, the resulting ranking function would first compute beliefs for american and #1(united states), then combine the beliefs using the probabilistic #or operator. The belief of this #or operator and the belief of presidents would be combined using the probabilistic #weight operator to produce the belief that the information need is satisfied. Within the inference network framework, queries are not explicitly defined. Instead, a structured query is used to construct a query network, which encodes a user's information need.

Finally, we note that, since language modeling probabilities are used within the network, the #wsum operator no longer makes sense. Instead, the #combine (#and) and #weight (#wand) operators are more appropriate, since they produce an *idf* effect (Metzler and Croft 2004). It can be shown that the Indri query #combine($q_1 \ldots q_N$) using the estimates just described returns exactly the same ranked list as the query $q_1 \ldots q_N$ using the traditional (multinomial with Dirichlet smoothing) query likelihood model.

3.2.5 Explicit vs. Implicit Query Generation

The Indri retrieval model can be used in two ways. First, given a simple keyword query, a system can be developed to convert the query into a structured Indri query. This process acts to transform the simple query into a richer representation. For example, phrases, synonyms, or task-specific operators may be automatically added to the query in order to improve effectiveness over the simple keyword query. The Indri retrieval model was successfully used in this capacity during the 2004–2006 TREC Terabyte Tracks (Metzler et al. 2004b, 2005b, 2006), and the 2005 TREC Robust Track (Metzler et al. 2005a).

Alternatively, users can use the query language to manually construct complex queries. It has been shown that intelligently constructed manual queries can significantly outperform automatically generated queries (Metzler and Croft 2004). However, the query language is too complex for novice users to use successfully. Only expert users, such as information analysts and librarians, are likely to benefit from such a query language. For this reason, algorithmic query construction is important.

Despite its success, the Indri retrieval model does not provide a formal mechanism for learning how to combine various types of evidence, making use of arbitrary evidence, or automatically converting a short keyword query into a rich structured query. The feature-based model that we present in this work is inspired by the Indri retrieval model, and attempts to overcome some of its limitations.

3.3 Other Models That Go Beyond the Bag of Words Assumption

There have been many models proposed to that go beyond the bag of words assumption (Clarke et al. 1995; Croft et al. 1991; Croft 1986; Fagan 1987; de Kretser and Moffat 1999). We now briefly highlight several of these models.

Fagan examines how to identify and use non-syntactic (statistical) phrases (Fagan 1987). Fagan identifies phrases using factors such as the number of times the phrase occurs in the collection and the proximity of the phrase terms. His results suggest no single method of phrase identification consistently yields improvements in retrieval effectiveness across a range of collections. For several collections, significant improvements in effectiveness are achieved when phrases are defined as any two terms within a query or document with unlimited proximity. That is, any two terms that co-occurred within a query or document were considered a phrase. However, for other collections, this definition proved to yield marginal or negative improvements.

Work done by Croft et al. shows similar results (Croft et al. 1991). Their results showed phrases formed with a probabilistic AND operator slightly outperformed proximity phrases. The probabilistic AND operator boosts the scores of documents where the phrase terms co-occur. Therefore, little benefit was shown as the result of modeling term proximity.

In addition to the n-gram language models we described, several other language model variants have been proposed that attempt to model term dependencies (Gao et

al. 2004; Nallapati and Allan 2002). The dependence language model presented by Gao et al. (2004) showed consistent improvements over a baseline query likelihood system on a number of TREC collections. However, the model uses a link structure for each query which is not straightforward to construct. Feature-based models do not require a query link structure to be constructed. However, if such information is available, it can easily be incorporated into such models.

Recently, Mishne and de Rijke explored the use of proximity information to improve Web retrieval (Mishne and de Rijke 2005). The Markov random field model shares many closely related insights. Despite the high level similarity, the details of the models differ greatly, with the Markov random field model allowing more general query dependencies and features to be considered in a more formally well-grounded framework.

Therefore, there have been many models proposed to go beyond the bag of words assumption, but none of them have allowed the use of arbitrary features, easy modeling of term dependencies, and yielded consistent, significant improvements in retrieval effectiveness. The model that we describe in the next chapter combines and generalizes the best aspects of these previous models within a robust, effective retrieval framework.

4 The Current State-of-the-Art

Despite the large number of attempts to go beyond the bag of words assumption, there have been very few, if any, models that have been proven to be consistently better than the current best bag of words models (i.e., language modeling and BM25).

In fact, strong evidence that BM25 and language modeling are considered the state-of-the-art retrieval models comes from looking at the models used by participants in recent years at TREC. Outside of several obscure, poor performing models, a majority of participant used either BM25 or language modeling, with some additional task-specific engineering added on top. Therefore, little progress has been made in advancing the state-of-the-art of retrieval models since the advent of language modeling and BM25 a decade ago.

Chapter 3
Feature-Based Ranking

1 Overview

In this chapter we introduce a feature-based retrieval model based on Markov random fields, which we refer to as the *Markov random field model for information retrieval* (MRF model). Although there are many different ways to formulate a general feature-based model for information retrieval, we focus our attention throughout this work on the Markov random field model because it satisfies the following desiderata, which we originally outlined in Chap. 1:

1. Supports basic information retrieval tasks (e.g., ranking, query expansion, etc.).
2. Easily and intuitively models query term dependencies.
3. Handles arbitrary textual and non-textual features.
4. Consistently and significantly improves effectiveness over bag of words models across a wide range of tasks and data sets.

Another reason we focus on the MRF model is because it has been the focus of a great deal of recent research and has been consistently shown to provide a robust, flexible, and extensible feature-based retrieva framework (Bendersky and Croft 2008; Eguchi 2005; Lang et al. 2010; Lease 2009; Metzler et al. 2004b, 2005b, 2006; Metzler and Croft 2005, 2007; Wang et al. 2010a, 2010b). Furthermore, there are a number of open source information retrieval toolkits, including Indri (Strohman et al. 2004) and Ivory (Lin et al. 2009), that include implementations of the MRF model and its various extensions, making it easy for researchers to experiment with the models.

The remainder of this chapter covers the basic theoretical and practical foundations of the model. Subsequent chapters will go into more detail and describe various extensions of the basic model.

2 Modeling Relevance

We begin by describing what we seek to model. The four primary variables in most information retrieval systems are users ($\mathcal{U}$), queries ($\mathcal{Q}$), documents ($\mathcal{D}$), and rele-

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8_3, © Springer-Verlag Berlin Heidelberg 2011

vance ($\mathcal{R}$). We define the event space to be $\mathcal{U} \times \mathcal{Q} \times \mathcal{D}$ and define relevance, $R \in \mathcal{R}$, to be a random variable over $\mathcal{U} \times \mathcal{Q} \times \mathcal{D}$. Thus, some relevance value is associated with every user, query, document tuple. Other factors, such as time and context are ignored.

These variables interact in real information systems in the following way. Users submit queries to the system and are presented a ranked list of documents. Some of the documents in the ranked are relevant, while others are non-relevant. Suppose that we were to collect a list of query/document pairs (Q, D), such that some user found document D relevant to query Q. Imagine that such a list was collected across a large sample of users. The resulting list can be thought of as a sample from some underlying population of relevant query/document pairs that are aggregated across users and conditioned on relevance. This, is then, a *relevance distribution*[1], which is similar in spirit to the one proposed by Lavrenko (2004). It is this distribution, $P(Q, D|R = 1)$, the joint distribution over query and document pairs, conditioned on relevance, that we focus on modeling. For notational convenience, we drop the explicit conditioning on relevance (i.e., $R = 1$) throughout the remainder of this work, unless otherwise noted.

3 The Markov Random Field Model

There are many possible ways to model a joint distribution. In this work, we choose to use Markov random fields. Markov random fields, sometimes referred to as undirected graphical models, are commonly used in the statistical machine learning domain to model complex joint distributions. As we will show throughout the remainder of this section, there are many advantages and few, if any, disadvantages to using MRFs for information retrieval.

A Markov random field is constructed from a graph G. The nodes in the graph represent random variables, and the edges define the independence semantics between the random variables. The independence semantics are governed by the Markov property.

> **Markov Property.** Let $G = (V, E)$ be the undirected graph associated with a Markov random field, then $P(v_i|v_{j\neq i}) = P(v_i|v_j : (v_i, v_j) \in E)$ for every random variable v_i associated with a node in V.

The Markov Property states that every random variable in the graph is independent of its non-neighbors given observed values for its neighbors. Therefore, different edge configurations impose different independence assumptions.

There are several ways to model the joint distribution $P(Q, D)$ using Markov random fields. Figure 3.1 summarizes the various options that are available. Option A constructs a graph with two nodes, a query node Q and a document node D.

[1] Note that we make the assumption that relevance is binary, which is commonly used for information retrieval tasks. If relevance is non-binary, then a different relevance distribution can be estimated for each relevance level.

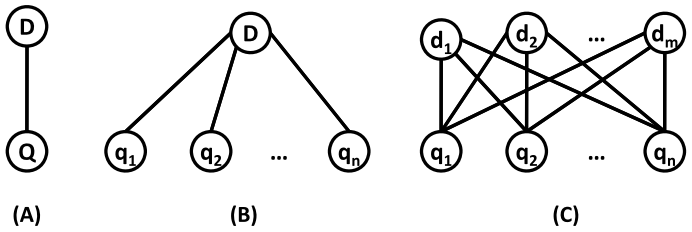

Fig. 3.1 Three possible ways to model the joint distribution $P(Q, D)$ using Markov random fields

However, this model is too coarsely specified and does not provide any insight into the types of term dependencies that are being modeled since it models whole queries and documents. Option B breaks the query apart into individual terms and treats the document as a whole. Given a query of length n, this results in a graph with n query term nodes and a document node. This option provides more specific control over which query term dependencies are modeled. Finally, option C breaks apart both the document and the query into individual terms. Given a query of length n and a document of length m, the graph would contain n query term nodes and m document term nodes. This option provides the most flexibility for modeling both query and document term dependencies. However, the model is likely to be overly complex. Modeling dependencies between query terms is more feasible than modeling dependencies between document terms since queries are generally much shorter than documents and exhibit less complex dependencies between terms.

Option B satisfies our needs without being overly complex, and so it will be used throughout the remainder of this work. Thus, given a query of length n, the graph G consists of n query term nodes and a single document node D. The random variables associated with the query term nodes are multinomials over the underlying vocabulary $\mathcal{V}$ and the random variable associated with the document node is also a multinomial over the set of documents in the collection. We note that variations on this theme are possible. For example, it may be appropriate to include several document nodes or even other types of nodes, such as document structure nodes within the MRF.

The joint probability mass function over the random variables in G is defined by:

$$P_{G,\Lambda}(Q, D) = \frac{1}{Z_\Lambda} \prod_{c \in C(G)} \psi(c; \Lambda), \tag{3.1}$$

where $Q = q_1 \ldots q_n$ are the set of query term nodes, D is the document node, $C(G)$ is the set of maximal cliques in G, each $\psi(\cdot; \Lambda)$ is a non-negative *potential function* over clique configurations parameterized by Λ and $Z_\Lambda = \sum_{Q,D} \prod_{c \in C(G)} \psi(c; \Lambda)$ normalizes the distribution. It is generally infeasible to compute Z_Λ due to the exponential number of terms in the summation.

Therefore, in order to compute the joint distribution we need a graph G, potential functions ψ, and the parameter vector Λ. Detailed descriptions of these components are given in the following sections.

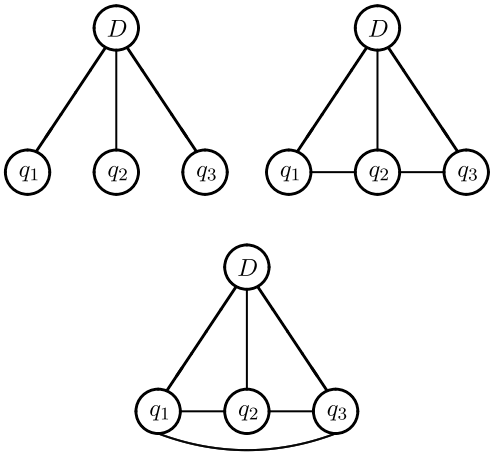

Fig. 3.2 Example Markov random field model for three query terms under various independence assumptions, including full independence (*top left*), sequential dependence (*top right*), and full dependence (*bottom*)

3.1 Graph Structure

We have already described how the nodes of the MRF are chosen. We now must show how these nodes can be connected together. As explained before, the Markov Property dictates the dependence semantics of the MRF. Therefore, it is relatively straightforward to explore various independence assumptions by constructing MRFs with different graph structures.

We consider three generalized graph structures, each with different underlying independence assumptions. The three structures are *full independence* (FI), *sequential dependence* (SD), and *full dependence* (FD). Figure 3.2 shows graphical model representations of each. These generalized structures are considered because of their significance to information retrieval. As we now show, each corresponds to a well-studied class of retrieval models.

The full independence structure makes the assumption that query terms q_i are independent given some document D. That is, the likelihood of observing query term q_i is not affected by the observation of any other query term, or more succinctly, $P(q_i|D, q_{j\neq i}) = P(q_i|D)$. This corresponds to the independence assumption made by many of the bag of words models that were described in Chap. 2.

As its name implies, the sequential dependence structure assumes a dependence between neighboring query terms. Formally, this assumption states that $P(q_i|D, q_{j\neq i}) = P(q_i|D, q_{i-1}, q_{i+1})$. Models of this form are similar in nature to bigram and biterm language models (Song and Croft 1999; Srikanth and Srihari 2002).

The last structure we consider is the full dependence structure. In this structure, we assume all query terms are in some way dependent on each other. Graphically, a query of length n translates into the complete graph K_{n+1}, which includes edges from each query node to the document node D, as well. This model is an attempt to capture longer range dependencies than the sequential dependence structure. If such a model can accurately be estimated, it should be expected to perform at least as well as a model that ignores term dependence.

There are other reasonable ways of constructing G given a query, such as that proposed by Gao et al. (2004), in which dependencies between terms are inferred using natural language processing techniques. The advantage of using one of the structures just described is that there is no need to rely on natural language processing techniques, which can often produce noisy output, especially on short segments of text. Of course, some of the dependencies imposed by the structure may be incorrect, but in general, they capture meaningful relationships between terms.

3.2 Potential Functions

In order to compute the MRF's joint probability mass function (Eq. 3.1), a set of potential functions must be defined over configurations of the maximal cliques in the underlying graph. These potential functions can be thought of as *compatibility functions*. That is, they are meant to reflect how compatible a given clique configuration is. How compatibility is defined and measured depends on the task and clique.

For example, in Fig. 3.2, the nodes D and q_1 form a maximal clique in the full independence variant. The potential function defined over the clique should reflect how compatible the term q_1 is to D. Here, compatibility may be defined as "aboutness" and measured using some *tf.idf* score for the term q_1 in D.

Typically, potential functions are built top-down, starting with a maximal clique and defining a potential over it. However, within the model, we choose to build potential functions in a bottom-up fashion, which provides more fine grained control over the behavior of the functions. This is accomplished by first associating one or more real-valued *feature functions* with each (maximal or non-maximal) clique in the graph. Each feature function has a feature weight associated with it that is a free parameter in the model. Then, non-negative potential functions over the maximal cliques are constructed from these feature functions and feature weights using an exponential form. We now formally describe the details of this process.

1. Assign one or more feature functions to each clique in G. This assignment can be encoded as a set of 3-tuples, $\mathcal{C} = \{(c, f(\cdot), \lambda)\}_{i=1}^{n}$, where c is a clique of G, $f(\cdot)$ is the feature function assigned to the clique, and λ is the weight (parameter) associated with the feature. Recall that the same clique may be associated with more than one feature function.
2. For every $(c, f(\cdot), \lambda) \in \mathcal{C}$, assign c to one of the maximal clique(s) in G that c is a sub-clique of. It is always possible to assign sub-cliques to maximal cliques, although this assignment is not guaranteed to be unique.
3. For every maximal clique in G, define its potential function as $\psi(\cdot) = \exp(\sum_c \lambda f(\cdot))$, where the sum goes over the cliques that were assigned to the maximal clique in Step 2.

We now provide an example to illustrate the process. Consider the full independence graph in Fig. 3.2. Suppose that we make the following assignment of feature

functions and parameters to the graph:

$$\begin{gathered}
\big(\{q_1, D\}, f_1(q_1, D), \lambda_1\big), \\
\big(\{q_1, D\}, f_4(q_1, D), \lambda_4\big), \\
\big(\{q_2, D\}, f_2(q_2, D), \lambda_2\big), \\
\big(\{q_2, D\}, f_4(q_2, D), \lambda_4\big), \\
\big(\{q_3, D\}, f_3(q_3, D), \lambda_3\big), \\
\big(\{q_3, D\}, f_4(q_3, D), \lambda_4\big), \\
\big(\{D\}, f_5(D), \lambda_5\big),
\end{gathered}$$

where each f_i is some real-valued feature function defined over the configurations of the clique. The specific form of the feature functions is not important in this example.

After assigning each clique to a maximal clique, we construct the following potential functions:

$$\psi(q_1, D) = \exp\big[\lambda_1 f_1(q_1, D) + \lambda_4 f_4(q_1, D) + \lambda_5 f_5(D)\big], \tag{3.2}$$

$$\psi(q_2, D) = \exp\big[\lambda_2 f_2(q_2, D) + \lambda_4 f_4(q_2, D)\big], \tag{3.3}$$

$$\psi(q_3, D) = \exp\big[\lambda_3 f_3(q_3, D) + \lambda_4 f_4(q_3, D)\big]. \tag{3.4}$$

This construction is not unique since the clique $\{D\}$ is a sub-clique of all three maximal cliques. Therefore, we can assign feature function $f_5(D)$ to any of the maximal cliques. In the previous set of potential functions, it was assigned to the maximal clique $\{q_1, D\}$. If it had been assigned to the maximal clique $\{q_3, D\}$ instead, the following potential functions would have been constructed:

$$\psi(q_1, D) = \exp\big[\lambda_1 f_1(q_1, D) + \lambda_4 f_4(q_1, D)\big], \tag{3.5}$$

$$\psi(q_2, D) = \exp\big[\lambda_2 f_2(q_2, D) + \lambda_4 f_4(q_2, D)\big], \tag{3.6}$$

$$\psi(q_3, D) = \exp\big[\lambda_3 f_3(q_3, D) + \lambda_4 f_4(q_3, D) + \lambda_5 f_5(D)\big]. \tag{3.7}$$

It is critical to note that, even though the potential function definitions are not guaranteed to be unique using this formulation, the joint probability mass function will be unique. It is easy to see that, for this example, the joint, under all possible assignments is equal to:

$$\begin{aligned}
P(Q, D) = Z_\Lambda^{-1} \exp\big[&\lambda_1 f_1(q_1, D) + \lambda_4 f_4(q_1, D) \\
&+ \lambda_2 f_2(q_2, D) + \lambda_4 f_4(q_2, D) \\
&+ \lambda_3 f_3(q_3, D) + \lambda_4 f_4(q_3, D) \\
&+ \lambda_5 f_5(D)\big].
\end{aligned} \tag{3.8}$$

This example also serves to illustrate that both functions and parameters can be shared across potential functions. Here, the feature function f_4 and parameter λ_4 were shared across three cliques. In order to share a feature function across cliques, we require that the input to the feature function be compatible with each clique. For example, a feature function that takes two term nodes and a document node as input can only be shared across cliques with two term nodes and a document node. We do not permit a feature function that only takes a document node as input to be shared with a clique that contains both a query term node and a document node. There are no restrictions on sharing parameters across cliques, however. By sharing parameters across cliques, we effectively tie parameters together, which reduces the number of free parameters and can help overcome data sparseness issues.

4 Constructing Markov Random Fields

As we just showed, potential functions are constructed by assigning feature functions and parameters to arbitrary cliques in the MRF. In this section, we describe how textual and non-textual features can be represented and assigned to cliques. Potentials can then be built from these features and be used to compute $P(Q, D)$.

In this section, we describe a method for representing MRFs for information retrieval. We represent MRFs using a *canonical form*. The canonical form is designed to be a compact, intuitive, and flexible method of representing MRFs. It can handle a wide variety of graph structures and features that are useful for information retrieval tasks. A canonical forms have the following structure:

$$\text{(dependence model type, clique set type, weighting function)}_1 : \lambda_1$$
$$\text{(dependence model type, clique set type, weighting function)}_2 : \lambda_2$$
$$\ldots$$
$$\text{(dependence model type, clique set type, weighting function)}_n : \lambda_n$$

Here, a 3-tuple represents how feature functions are assigned to cliques. Each 3-tuple assigns a feature function to one or more cliques within the graph. The variable after the colon represents the parameter associated with all of the feature functions assigned by the 3-tuple. As we showed in the previous section, this ties the parameters of all of the feature functions associated with a 3-tuple together. The details of this assignment and tying process will become clearer later in this section when we work through several examples.

Given a canonical form, it is easy to systematically build the corresponding MRF and derive both the joint probability mass function, as well as the ranking function. We represent all MRFs throughout the remainder of this work using these canonical forms. We now describe the meaning and details of each component in the 3-tuple.

4.1 Dependence Model Type

The first entry in the tuple is the *dependence model type*, which specifies the dependencies, if any, that are to be modeled between query terms. As we described before, dependencies are encoded by the edges in the MRF, with different edge configurations correspond to different types of dependence assumptions.

In this work, we only allow the dependence model type to be full independence (FI), sequential dependence (SD), or full dependence (FD), which are the three generalized graph structures described in Sect. 3.1 and illustrated in Fig. 3.2.

For a given MRF, each feature function may have a different dependence model type. The dependence model type simply defines the graph structure that the current feature is applied to. The graph structure that the resulting MRF has depends on the dependence model types of all of its features combined.

4.2 Clique Set Type

The second entry in the tuple, the *clique set type*, describes the set of (maximal or non-maximal) cliques within the graph that the feature function is to be applied to. Thus, each feature function can be applied to one or more cliques within the graph, depending on the clique set.

There are seven clique sets that can be used within the model. These sets are summarized in Table 3.1. In order to motivate these clique sets, we enumerate every possible type of clique that is of interest to us, beginning with cliques that contain the document node and one or more query term nodes.

First, the simplest type of clique that contains the document node and one or more query nodes is a 2-clique consisting of an edge between a query term q_i and the document D. A potential function over such a clique should measure how well, or how likely query term q_i describes the document.

Next, we consider cliques that contain two or more query terms. For such cliques there are two possible cases, either all of the query terms within the clique appear contiguously in the query or they do not. The fact that query terms appear contiguously within a query provides different (stronger) evidence about the information need than a set of non-contiguous query terms. For example, in the query *train station security measures*, if any of the sub-phrases, *train station*, *train station security*, *station security measures*, or *security measures* appear in a document then there is strong evidence in favor of relevance.

Although the occurrence of contiguous sets of query terms provide strong evidence of relevance, it is also the case that the occurrence of non-contiguous sets of query terms can provide valuable evidence. However, since the query terms are not contiguous we do not expect them to appear in order within relevant documents. Rather, we only expect the terms to appear ordered or unordered within a given proximity of each other. In the previous example, documents containing the terms

Table 3.1 Example clique sets for the query $q1\ q2\ q3$ under full dependence model

Description	Notation	Example
Set of cliques containing the document node and exactly one query term	T_{QD}	$\{\{q_1, D\}, \{q_2, D\}, \{q_3, D\}\}$
Set of cliques containing the document node and two or more query terms that appear in sequential order within the query	O_{QD}	$\{\{q_1, q_2, D\}, \{q_2, q_3, D\}, \{q_1, q_2, q_3, D\}\}$
Set of cliques containing the document node and two or more query terms that appear unordered within the query	U_{QD}	$\{\{q_1, q_3, D\}\}$
Set of cliques containing exactly one query term	T_Q	$\{\{q_1\}, \{q_2\}, \{q_3\}\}$
Set of cliques containing two or more query terms that appear in sequential order within the query	O_Q	$\{\{q_1, q_2\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$
Set of cliques containing two or more query terms that appear unordered within the query	U_Q	$\{\{q_1, q_3\}\}$
Set containing only the singleton node D	D	$\{\{D\}\}$

train and *security* within some short proximity of one another also provide additional evidence towards relevance. This issue has been explored in the past by a number of researchers (Croft et al. 1991; Fagan 1987).

Therefore, for cliques consisting of the document node and one or more query term nodes, we have the following clique sets:

- T_{QD}—set of cliques containing the document node and exactly one query term.
- O_{QD}—set of cliques containing the document node and two or more query terms that appear in sequential order within the query.
- U_{QD}—set of cliques containing the document node and two or more query terms that appear unordered within the query.

Note that the cliques that make up each set may change for different dependence model types. For example, O_{QD} and U_{QD} are empty under the full independence assumption since that would result in a graph where there are no cliques with two or more query term nodes. However, under the sequential dependence assumption, and with a query of length 2 or more, such cliques will exist and O_{QD} and U_{QD} will be non-empty.

Next, we consider cliques that only contain query term nodes. These clique sets are defined in an analogous way to those just defined, except the cliques are only made up of query term nodes and do not contain the document node. Potential functions over these cliques should capture how compatible query terms are to one

another. These clique potentials may take on the form of language models that impose well-formedness of the terms. Therefore, we define following query-dependent clique sets:

- T_Q—set of cliques containing exactly one query term.
- O_Q—set of cliques containing two or more query terms that appear in sequential order within the query.
- U_Q—set of cliques containing two or more query terms that appear unordered within the query.

Finally, there is the clique that only contains the document node. Potentials over this node can be used as a type of document prior, encoding document-centric properties. This trivial clique set is then:

- D—clique set containing only the singleton node D.

We note that the clique sets form a partition over the cliques of G. This partition separates the cliques into sets that are meaningful from an information retrieval perspective. Thus, these clique sets make it easy to apply features in a very specific manner within the MRF.

Of course, the clique sets we defined here are not unique. It is possible to define many different types of clique sets. For example, another clique set may be defined as "the clique that contains the first query term and the document node". Given enough training data, it may be possible to define such fine grained clique sets. However, given the limited amount of training data, we focus our attention on the coarse grained clique sets defined above.

4.3 Weighting Function

Finally, the third entry in the tuple is the *weighting function*, which defines the feature function that is applied to the cliques defined by the clique set. In this section we define weighting functions that can be used with the different clique sets we just defined. It is not our goal to provide a comprehensive list of possible feature functions. Instead, we simply seek to provide a few examples of the types of feature functions that are possible.

4.3.1 Weighting Functions for T_{QD}, O_{QD}, and U_{QD}

We first describe weighting functions that can be used with cliques in the T_{QD}, O_{QD}, and U_{QD} clique sets. These cliques consist of a set of query term nodes and a document node. Therefore, the weighting functions applied to these cliques should measure how much the document is "about" the query terms.

The weighting functions we use are based on language modeling estimates and the BM25 weighting model, which we described in Chap. 2. It is straightforward

to use the standard forms for these weighting functions for the single term cliques (T_{QD}). However, we must define how to match the query terms within documents when applying these weighting functions to ordered term cliques (O_{QD}) and unordered term cliques (U_{QD}).

For ordered term cliques, we match terms in documents using the Indri ordered window operator (#M), where the parameter M determines how many non-matching terms are allowed to appear between matched terms (Metzler and Croft 2004). For clique $\{q_i, \ldots, q_{i+k}, D\}$, we match documents according to #M($q_i \ldots q_{i+k}$). This rewards documents for preserving the order that the query terms occur in.

In the unordered clique set case, we match terms using the Indri unordered window operator (#uw$N$), where N defines the maximum size of the window that the terms may occur (ordered or unordered) in. For clique $\{q_i, \ldots, q_j, D\}$ that contains k query terms, documents are matched according to #uw$Nk(q_i \ldots q_j)$. Notice that we multiply the number of terms in the clique set by N. If $N = 1$, then all k query terms must occur, ordered or unordered, within a window of k terms of each other within the document. As N increases, the matching becomes looser. If $N = unlimited$, then any document that contains all k query terms is matched. By using this matching scheme, we reward documents in which subsets of query terms occur appear within close proximity of each other.

Table 3.2 summarizes these weighting functions. Of course, many different types of weighting functions can easily be used within the model. For example, if new, more effective term weighting functions are developed in the future, then they can be easily used instead of, or in addition to, the Dirichlet or BM25 weighting functions.

4.3.2 Weighting Functions for T_Q, O_Q, and U_Q

Next, we consider weighting functions for the cliques in the T_Q, O_Q, and U_Q clique sets. These cliques consist of one or more query terms and no document nodes. Weighting functions defined over them should reflect their general importance or informativeness. Therefore, IDF-based measures are a natural set of feature functions to use for these types of cliques.

The two IDF measures that are used as feature functions are inverse collection frequency (ICF) and the Okapi IDF. Inverse collection frequency is very similar to IDF, except it considers the number of times an expression occurs, rather than the number of documents it occurs in. As with the weighting functions described in the previous section, it is straightforward to apply standard IDF features to the single term cliques (T_Q). We use the same matching semantics as described in the previous section for the ordered terms cliques (O_Q) and the unordered terms cliques (U_Q).

Example feature functions are shown in Table 3.3. Other possible feature functions for these types of cliques include measures of how lexically cohesive the terms are and the average vocabulary level of the terms.

Table 3.2 Summary of Dirichlet and BM25 weighting functions that can be used with cliques in the T_{QD}, O_{QD}, and U_{QD} clique sets. Here, M and N act as weighting function parameters that affect how matching is done, $tf_{e,D}$ is the number of times expression e matches in document D, $cf_{e,D}$ is the number of times expression e matches in the entire collection, df_e is the total number of documents that have at least one match for expression e, $|D|$ is the length of document D, $|D|_{\text{avg}}$ is the average document length, N is the number of documents in the collection, and $|C|$ is the total length of the collection. Finally, $idf(e) = \log \frac{N - df_e + 0.5}{df_e + 0.5}$, and μ^t, μ^w, k_1^t, k_1^w, b^t, and b^w are weighting function hyperparameters. The t and w superscripts indicate term and window hyperparameters, respectively

LM

$$f_{LM,T}(q_i, D) = \log\left[\frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t}\right]$$

LM-O-M

$$f_{LM,O,M}(\{q_i\}, D) = \log\left[\frac{tf_{\#M(\{q_i\}),D} + \mu^w \frac{cf_{\#M(\{q_i\})}}{|C|}}{|D| + \mu^w}\right]$$

LM-U-N

$$f_{LM,U,N}(\{q_i\}, D) = \log\left[\frac{tf_{\#\text{uw}Nk(\{q_i\}),D} + \mu^w \frac{cf_{\#\text{uw}Nk(\{q_i\})}}{|C|}}{|D| + \mu^w}\right]$$

BM25

$$f_{T,\text{BM25}}(q_i, D) = \frac{(k_1^t + 1) tf_{w,D}}{k_1^t((1 - b^t) + b^t \frac{|D|}{|D|_{\text{avg}}}) + tf_{w,D}} idf(w)$$

BM25-O-M

$$f_{\text{BM25},O,M}(\{q_i\}, D) = \frac{(k_1^w + 1) tf_{\#M(\{q_i\}),D}}{k_1^w((1 - b^w) + b^w \frac{|D|}{|D|_{\text{avg}}}) + tf_{\#M(\{q_i\}),D}} idf(\#M(\{q_i\}))$$

BM25-U-N

$$f_{\text{BM25},U,N}(\{q_i\}, D) = \frac{(k_1^w + 1) tf_{\#\text{uw}Nk(\{q_i\}),D}}{k_1^w((1 - b^w) + b^w \frac{|D|}{|D|_{\text{avg}}}) + tf_{\#\text{uw}Nk(\{q_i\}),D}} idf(\#\text{uw}Nk(\{q_i\}))$$

4.3.3 Weighting Functions for D

Depending on the task, there are a wide variety of weighting functions that can be applied to the document node clique. Some examples include document length (Singhal et al. 1996), document quality (Zhou and Croft 2005), PageRank (Brin and Page 1998), URL depth (Kraaij et al. 2002), readability (Si and Callan 2001), sentiment (Pang et al. 2002), and opinionatedness (Ounis et al. 2006).

Although we do not explore all of these query independent features in this work, we do make use of several of them for a Web search task later in this chapter.

Table 3.3 Summary of ICF and IDF weighting functions that can be used with cliques in the T_Q, O_Q, and U_Q clique sets

ICF

$$f_{\text{ICF},T}(q_i, D) = -\log \frac{cf_{q_i}}{|C|}$$

ICF-O-*M*

$$f_{\text{ICF},U,M}(\{q_i\}, D) = -\log \frac{cf_{\#M(\{q_i\})}}{|C|}$$

ICF-U-*N*

$$f_{\text{ICF},O,N}(\{q_i\}, D) = -\log \frac{cf_{\#\text{uwN}k(\{q_i\})}}{|C|}$$

IDF

$$f_{\text{IDF,BM25}}(q_i, D) = \log \frac{N - df_w + 0.5}{df_w + 0.5}$$

IDF-O-*M*

$$f_{\text{IDF},O,M}(\{q_i\}, D) = \log \frac{N - df_{\#M(\{q_i\})} + 0.5}{df_{\#M(\{q_i\})} + 0.5}$$

IDF-U-*N*

$$f_{\text{IDF},U,N}(\{q_i\}, D) = \log \frac{N - df_{\#\text{uwN}k(\{q_i\})} + 0.5}{df_{\#\text{uwN}k(\{q_i\})} + 0.5}$$

4.4 Examples

Now that we have described each element that makes up the 3-tuple, we show how to construct MRFs from canonical forms. We do this by working through a number of examples. In all of the following examples, it is assumed that the query being evaluated is *new york city*.

Our first example is for the following canonical form:

$$(\text{FI}, T_{QD}, \text{BM25}) : \lambda.$$

This canonical form includes a single feature function. The feature uses the full independence graph structure, is applied to the cliques in T_{QD}, and uses the BM25 weighting function. This expands to the following assignment of feature functions:

$$\big(\{\text{new}, D\}, f_{\text{BM25},T}(\text{new}, D), \lambda\big),$$
$$\big(\{\text{york}, D\}, f_{\text{BM25},T}(\text{york}, D), \lambda\big),$$
$$\big(\{\text{city}, D\}, f_{\text{BM25},T}(\text{city}, D), \lambda\big).$$

Notice that all of the features share the same parameter.

This assignment can then be transformed into the following set of potential functions, using the process described in Sect. 3.2:

$$\psi(\text{new}, D) = \exp\big[\lambda f_{\text{BM25},T}(\text{new}, D)\big], \tag{3.9}$$

$$\psi(\text{york}, D) = \exp\left[\lambda f_{\text{BM25},T}(\text{york}, D)\right], \tag{3.10}$$

$$\psi(\text{city}, D) = \exp\left[\lambda f_{\text{BM25},T}(\text{city}, D)\right], \tag{3.11}$$

where $f_{\text{BM25},T}$ takes on the BM25 form as given in Table 3.2. The resulting probability mass function is then given by:

$$P(\text{new york city}, D) = Z_{\Lambda}^{-1} \exp\left[\lambda f_{\text{BM25},T}(\text{new}, D) + \lambda f_{\text{BM25},T}(\text{york}, D) + \lambda f_{\text{BM25},T}(\text{city}, D)\right]. \tag{3.12}$$

We see that this joint probability mass function is rank equivalent to the BM25 score of query for document D. Analogously, if $f_{\text{BM25},T}$ is replaced with $f_{\text{LM},T}$, the probability mass function is rank equivalent to query likelihood scoring in the language modeling framework.

Next, we consider the following canonical form:

$$(\text{SD},\ O_{QD},\ \text{LM-O-4})\ :\ \lambda$$

which contains a single feature that uses the sequential dependence model, is applied to cliques in O_{QD}, and uses the Dirichlet weighting function. This expands into the following assignment of feature functions to cliques:

$$\left(\{\text{new}, \text{york}, D\},\ f_{\text{LM},O,4}(\text{new}, \text{york}, D),\ \lambda\right),$$
$$\left(\{\text{york}, \text{city}, D\},\ f_{\text{LM},O,4}(\text{york}, \text{city}, D),\ \lambda\right)$$

which is then transformed into the following set of potential functions:

$$\psi(\text{new}, \text{york}, D) = \exp\left[\lambda f_{\text{LM},O,4}(\text{new}, \text{york}, D)\right], \tag{3.13}$$

$$\psi(\text{york}, \text{city}, D) = \exp\left[\lambda f_{\text{LM},O,4}(\text{york}, \text{city}, D)\right], \tag{3.14}$$

where $f_{\text{LM},O,4}$ takes on the Dirichlet form and M, the ordered window size, is set to 4.

Finally, we provide an example of a more complex canonical form. Consider the following canonical form:

$$(\text{FD},\ O_{QD},\ \text{LM-O-8})\ :\ \lambda_1,$$
$$(\text{FI},\ T_Q,\ \text{IDF})\ :\ \lambda_2,$$
$$(\text{FI},\ D, \text{PageRank})\ :\ \lambda_3$$

which then results in the following set of feature function assignments:

$$\left(\{\text{new}, \text{york}, D\},\ f_{\text{LM},O,8}(\text{new}, \text{york}, D),\ \lambda_1\right),$$
$$\left(\{\text{york}, \text{city}, D\},\ f_{\text{LM},O,8}(\text{york}, \text{city}, D),\ \lambda_1\right),$$
$$\left(\{\text{new}, \text{york}, \text{city}, D\},\ f_{\text{LM},O,8}(\text{new}, \text{york}, \text{city}, D),\ \lambda_1\right),$$

$$\big(\{\text{new}\},\ f_{\text{IDF},T}(\text{new}, D),\ \lambda_2\big),$$
$$\big(\{\text{york}\},\ f_{\text{IDF},T}(\text{york}, D),\ \lambda_2\big),$$
$$\big(\{\text{city}\},\ f_{\text{IDF},T}(\text{city}, D),\ \lambda_2\big),$$
$$\big(\{D\},\ f_{\text{PageRank}}(D),\ \lambda_3\big)$$

and the following potential function:

$$\begin{aligned}\psi(\text{new, york, city}, D) = \exp\big[&\lambda_1 f_{\text{LM},O,8}(\text{new, york}, D)\\ &+ \lambda_1 f_{\text{LM},O,8}(\text{york, city}, D)\\ &+ \lambda_1 f_{\text{LM},O,8}(\text{new, york, city}, D)\\ &+ \lambda_2 f_{\text{IDF},T}(\text{new}, D)\\ &+ \lambda_2 f_{\text{IDF},T}(\text{york}, D)\\ &+ \lambda_2 f_{\text{IDF},T}(\text{city}, D)\\ &+ \lambda_3 f_{\text{PageRank}}(D)\big].\end{aligned} \tag{3.15}$$

These examples illustrate that the canonical form allows us to compactly define a large, rich set of MRFs for use with information retrieval tasks.

5 Ranking

Using the canonical feature representation, we derive the following simplified form of the joint distribution:

$$\begin{aligned}\log P(Q, D) = &\underbrace{\sum_{c\in T_{QD}} \lambda_c f_c(c) + \sum_{c\in O_{QD}} \lambda_c f_c(c) + \sum_{c\in U_{QD}} \lambda_c f_c(c)}_{\text{document + query dependent}}\\ &+ \underbrace{\sum_{c\in T_Q} \lambda_c f_c(c) + \sum_{c\in O_Q} \lambda_c f_c(c) + \sum_{c\in U_Q} \lambda_c f_c(c)}_{\text{query dependent}}\\ &+ \underbrace{\sum_{c\in D} \lambda_D f_c(c)}_{\text{document dependent}} - \underbrace{\log Z_\Lambda}_{\text{document + query independent}},\end{aligned} \tag{3.16}$$

where λ_c and f_c are the parameter and weighting (feature) function associated with clique c, respectively.

Given a query Q as evidence, we can use the model to rank documents in descending order according of the conditional $P(D|Q)$. Fortunately, properties of

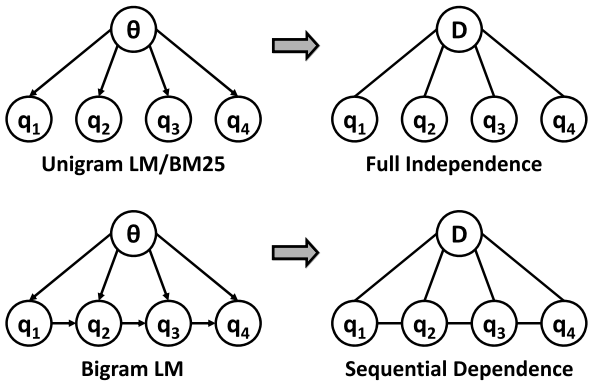

Fig. 3.3 Illustration showing how the full independence model generalizes unigram language modeling and BM25 (*top*), and how the sequential dependence model generalizes bigram language modeling (*bottom*)

rankings allow us to significantly simplify the computation. That is,

$$\begin{aligned} P(D|Q) &\overset{\text{rank}}{=} \log P(D|Q) \\ &= \log \frac{P(Q,D)}{P(Q)} \\ &= \log P(Q,D) - \log P(Q) \\ &\overset{\text{rank}}{=} \log P(Q,D). \end{aligned} \tag{3.17}$$

After dropping document independent expressions from $\log P(Q, D)$, we derive the following ranking function:

$$P(D|Q) \overset{\text{rank}}{=} \sum_{c \in T_{QD}} \lambda_c f_c(c) + \sum_{c \in O_{QD}} \lambda_c f_c(c) + \sum_{c \in U_{QD}} \lambda_c f_c(c) + \sum_{c \in D} \lambda_c f_c(c) \tag{3.18}$$

which is a simple weighted linear combination of feature functions that can be computed efficiently for reasonable graphs since the partition function Z_Λ does not need to be computed. Later, in Chap. 4 we show how the reverse conditional, $P(Q|D)$, can be used for query expansion.

In this chapter, we described the basics of the Markov random field model for information retrieval. We explained our underlying model of relevance, basic MRF theory, and how MRF models can easily be constructed using a canonical form. The model is very robust, as it can model a wide variety of dependencies between query terms, and can make use of arbitrary textual and non-textual features, as well. This is the first model for information retrieval that has both of these important properties.

It is easy to show that the MRF model subsumes many previously proposed information retrieval models, which further proves the model's flexibility. Figure 3.3 shows two simple examples of how the MRF model generalizes other models. Here, we see that the full independence model, with properly defined potential functions, gives rise to unigram language modeling and the BM25 model. Similarly, the sequential dependence model subsumes bigram and biterm language models.

By studying previous retrieval models in the context of the MRF model, we gain fresh perspective and insight into the underlying principles of these models.

This chapter skirted the issue of parameter estimation (i.e., how to set Λ). Since this issue is critical to achieving good effectiveness, it is given a detailed treatment in Chap. 6.

6 Ad Hoc Retrieval

Our discussion, up until this point, has focused entirely on the theoretical issues surrounding the Markov random field model. We now shift our focus to more practical matters. In this chapter, we empirically evaluate the retrieval effectiveness of the MRF model. This requires us to choose one or more tasks to evaluate the model against. There are a large number of important information retrieval tasks, such as Web search (Brin and Page 1998), enterprise search (Craswell et al. 2005a), question answering (QA) (Voorhees 1999), blog search (Ounis et al. 2006), legal search (Baron et al. 2006), desktop search (Peng and He 2006), and image search. Rather than evaluating the model on all of these tasks, we restrict our focus to *ad hoc* retrieval and Web search. As we will describe in more detail shortly, these two tasks are the most common and widely used in information retrieval.

Ad hoc retrieval is one of the most important information retrieval tasks. In the task, a user submits a query, and the system returns a ranked list of documents that are *topically* relevant to the query. Therefore, the goal of the task is to find topically relevant documents in response to a query. It is critical to develop highly effective *ad hoc* retrieval models since such models often play important roles in other retrieval tasks. For example, most QA systems use an *ad hoc* retrieval system to procure documents that are topically relevant to some question. The QA systems then employ various techniques to extract answers from the document retrieved (Voorhees 1999). Thus, by improving on the current state-of-the-art *ad hoc* retrieval models, it is possible to positively impact the effectiveness of a wide range of tasks.

In the remainder of this section we describe experiments using three different basic MRF models. The aim is to analyze and compare the retrieval effectiveness of each model across collections of varying size and type. We make use of the AP, WSJ, and ROBUST04 data sets, which are smaller collections that consist of news articles that are mostly homogeneous, and two Web data sets, WT10g and GOV2, which are considerably larger and less homogeneous. Further details about the data sets are provided in Appendix A.

Each of these are TREC data sets. A TREC data set consists of a collection of documents, a set of topics, and human relevance assessments. An example *ad hoc* topic is shown in Fig. 3.4. A TREC topic typically consists of a title, description, and narrative. It is important to note that a topic is not the same thing as a query, although the two terms are often used interchangeably. A query must be distilled from a topic. This is typically done by using the text contained in one or more of the topic fields as the query. For all of the experiments in this section, except where noted otherwise, we follow the common TREC procedure of using only the title

<top>
<num> Number: 744

<title>
Counterfeit ID punishments

<desc> Description:
What punishments or sentences have been given in the U.S. for making or selling counterfeit IDs?

<narr> Narrative:
Relevant documents will describe punishments for manufacturing or selling counterfeit identification, such as drivers licenses, passports, social security cards, etc. Fake professional certifications and fake credit cards are relevant. Counterfeit goods or auto serial numbers not relevant. Counterfeit checks are not relevant. "Counterfeiting" with no indication of type is relevant.

</top>

Fig. 3.4 TREC topic number 744

portion of the topic as the query. Therefore, the query that we distill for the topic given in Fig. 3.4 is *counterfeit id punishments*.

TREC relevance judgments are done by human assessors. When determining relevance, the entire TREC topic is considered. The assessors judge documents using a binary[2] scale, where rating 0 indicates not relevant and rating 1 indicates relevant.

All of the evaluation metrics that we consider in this section are based on binary judgments. For all of the experiments, we return a ranked list of no more than 1000 documents per query, as is standardly done during TREC evaluations. Furthermore, the primary evaluation metric that we use to evaluate *ad hoc* retrieval is mean average precision. Further details about the retrieval metrics we use can be found in Appendix B.

Finally, for all of the experiments, documents were stemmed using the Porter stemmer and a standard list of 418 stopwords was applied. All model parameters were estimated by maximizing mean average precision using a coordinate ascent algorithm (see Algorithm 2).

Throughout all of the experiments, statistical significance is always tested using a one-tailed, paired t-test at significance level $p < 0.05$.

[2] Although some TREC collections actually do have ternary (i.e., not relevant, relevant, and highly relevant) judgments, they have never been used during official evaluations. When ternary judgments do exist, all relevant (rating 1) and highly relevant (rating 2) documents are considered relevant, which thereby binarizes the judgments.

6.1 MRF Models for Ad Hoc Retrieval

We now define the three basic MRF models for the *ad hoc* retrieval task. The models correspond to the three dependence model types shown in Fig. 3.2. Each model represents a different set of underlying dependence assumptions and makes use of different features. We define each model in terms of its canonical form, provide its joint probability mass function, and show its ranking function.

6.1.1 Full Independence

The first basic model that we consider makes use of the full independence model shown in Fig. 3.2 (left). Recall that, under this model, query term nodes are independent of each other given a document as evidence. This model, therefore, shares many properties with standard bag of words retrieval models.

We now introduce the first basic MRF model, which we call the *Full Independence MRF Model* (MRF-FI model). It is constructed using the following canonical form:

$$(\text{FI}, T_{QD}, \text{LM}) \; : \; \lambda_{T_D},$$
$$(\text{FI}, T_Q, \text{ICF}) \; : \; \lambda_{T_Q}.$$

The model defines two features. One feature is defined over the T_{QD} clique set and the other is defined over the T_Q clique set. Both features use the full independence assumption and language modeling features.

Notice that no feature is defined over D, the document node clique set. By not defining a feature over the document node clique, we are enforcing the constraint that documents, in isolation, provide no useful information for the *ad hoc* retrieval task. While this may seem like an extreme assumption, it is actually quite valid. No single document prior has ever been shown to significantly improve effectiveness across a wide range of data sets. Therefore, in order to keep the model as simple as possible, we simply do not define a feature over this clique. However, we note that for specific tasks it may be beneficial to define such a feature.

The model results in the following joint probability mass function:

$$P(Q, D) = Z^{-1} \exp\Bigg[\lambda_{T_D} \sum_{(q_i, D) \in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t} + \lambda_{T_Q} \sum_{q_i \in T_Q} \log \frac{|C|}{cf_{q_i}}\Bigg]. \tag{3.19}$$

Table 3.4 Test set results for the MRF-FI model

	AP	WSJ	ROBUST04	WT10G	GOV2
MAP	0.2077	0.3258	0.2920	0.1861	0.2984
GMAP	0.1219	0.2267	0.1970	0.1176	0.1891
P@10	0.3460	0.4860	0.4293	0.3204	0.5180
R-Prec	0.2448	0.3558	0.3291	0.2199	0.3515
μ^t	1750	2000	1000	1000	1500

Furthermore, it is easy to see that we obtain the following linear feature-based model when ranking according to $P(D|Q)$:

$$P(D|Q) \stackrel{\text{rank}}{=} \sum_{(q_i,D)\in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t} \tag{3.20}$$

which shows that the MRF-FI model reduces exactly to the unigram query likelihood language modeling approach with Dirichlet smoothing (see Eq. 2.14).

Although the MRF-FI model is not technically a bag of words model, we consider it as a bag of words baseline. This is appropriate, since, as we just showed, the model is rank equivalent to the unigram language modeling approach, which is a bag of words model. Therefore, we use the MRF-FI model as a baseline by which we compare other MRF models that actually go beyond the bag of words assumption.

Table 3.4 shows the test set results for the MRF-FI model across data sets. In the table, MAP refers to mean average precision, GMAP is geometric mean average precision, P@10 is precision at 10 ranked documents, R-Precision is precision at R (number of judged relevant documents), and μ^t denotes the smoothing parameter learned on the training set. All models were trained to maximize mean average precision. These numbers serve as the baselines, which we attempt to improve upon by employing more complex models.

6.1.2 Sequential Dependence

The second of the basic MRF models corresponds to the sequential dependence model shown in Fig. 3.2 (center). It is the *Sequential Dependence MRF Model* (MRF-SD model), which is constructed according to the following canonical form:

$$(\text{FI}, T_{QD}, \text{LM}) \; : \; \lambda_{T_D},$$
$$(\text{FI}, T_Q, \text{ICF}) \; : \; \lambda_{T_Q},$$
$$(\text{SD}, O_{QD}, \text{LM-O-1}) \; : \; \lambda_{O_D},$$
$$(\text{SD}, O_Q, \text{ICF-O-1}) \; : \; \lambda_{O_Q},$$

$$(\text{SD}, O_{QD}, \text{LM-U-4}) : \lambda_{U_D},$$
$$(\text{SD}, O_Q, \text{ICF-U-4}) : \lambda_{U_Q}$$

which defines features over single term (i.e., T_{QD} and T_Q) clique sets, as well as ordered term clique sets (i.e., O_{QD} and O_Q). Unlike the MRF-SI model, this model makes use of some of the MRF model's strengths. As we see, the model defines ordered and unordered window features over the ordered cliques in the graph. By doing so, we go beyond the bag of words assumption. The joint probability mass function for the model is:

$$\begin{aligned}
P(Q, D) \propto \exp\Bigg[& \lambda_{T_D} \sum_{(q_i, D) \in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t} \\
& + \lambda_{T_Q} \sum_{q_i \in T_Q} \log \frac{|C|}{cf_{q_i}} \\
& + \lambda_{O_D} \sum_{(q_1, q_2, D) \in O_{QD}} \log \frac{tf_{\#1(q_1 q_2),D} + \mu^w \frac{cf_{\#1(q_1 q_2)}}{|C|}}{|D| + \mu^w} \\
& + \lambda_{O_Q} \sum_{(q_1, q_2) \in O_Q} \log \frac{|C|}{cf_{\#1(q_1 q_2)}} \\
& + \lambda_{U_D} \sum_{(q_1, q_2, D) \in U_{QD}} \log \frac{tf_{\#\text{uw8}(q_1 q_2),D} + \mu^w \frac{cf_{\#\text{uw8}(q_1 q_2)}}{|C|}}{|D| + \mu^w} \\
& + \lambda_{U_Q} \sum_{(q_1, q_2) \in U_Q} \log \frac{|C|}{cf_{\#\text{uw8}(q_1 q_2)}} \Bigg]
\end{aligned} \tag{3.21}$$

and the ranking function simplifies to the following linear feature-based model:

$$\begin{aligned}
P(D|Q) \overset{\text{rank}}{=} & \lambda_{T_D} \sum_{(q_i, D) \in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t} \\
& + \lambda_{O_D} \sum_{(q_1, q_2, D) \in O_{QD}} \log \frac{tf_{\#1(q_1 q_2),D} + \mu^w \frac{cf_{\#1(q_1 q_2)}}{|C|}}{|D| + \mu^w} \\
& + \lambda_{U_D} \sum_{(q_1, q_2, D) \in U_{QD}} \log \frac{tf_{\#\text{uw8}(q_1 q_2),D} + \mu^w \frac{cf_{\#\text{uw8}(q_1 q_2)}}{|C|}}{|D| + \mu^w}.
\end{aligned} \tag{3.22}$$

Table 3.5 Mean average precision for various parameter settings for LM-U-N using the MRF-SD model

N	AP	WSJ	WT10g	GOV2
2	0.1860	0.2776	0.2148	0.2697
8	0.1867	0.2763	0.2167	0.2832
50	0.1858	0.2766	0.2154	0.2817
Unlimited	0.1857	0.2759	0.2138	0.2714

Recall that both the ordered (LM-O-M) and unordered (LM-U-N) features have free parameters that allow the size of the unordered window (scope of proximity) to vary. We now motivate why these specific values were chosen.

First, for M, the parameter that controls the ordered window matching, we decided to use 1, because it results in an "exact phrase" feature that does not allow any room in the ordered matching of query terms. This choice is motivated by the fact that exact phrases are commonly used in many different applications. Furthermore, there has been little previous research that looked at relaxing such phrases. Therefore, choosing 1 is the most reasonable choice.

The other value, N, which controls the window width for unordered matching, was chosen after careful consideration of previous research. Fagan shows that the best choice of N varies across collections (Fagan 1987). Optimal values found included setting N to either 2, the length of a sentence, or "unlimited" (matches any co-occurrences of the terms within a document). Croft et al. showed improvements could be achieved with passage-sized windows of 50 terms (Croft et al. 1991). Therefore, since there were no strong conclusions, we experimented with window sizes of 2, 50, sentence, and "unlimited" to see what impact each had on effectiveness. Instead of segmenting sentences at index time, we observe that the average length of an English sentence is 8–15 terms, and choose a window size of 8 terms to model sentence-level proximity.

The results, which were evaluated on the entire data set, are given in Table 3.5. The results show very little difference across the various window sizes. However, for the AP, WT10g, and GOV2 collection the sentence-sized windows performed the best. For the WSJ collection, $N = 1$ performed the best. The only collection where mean average precision varies noticeably is the GOV2 collection. These results suggest that a limited scope of proximity (2–50 terms) performs reasonably, but can be approximated rather well by an "unlimited" scope, which reaffirms past research into dependence models based on co-occurrences. However, it appears as though smaller scopes of proximity may provide better performance for larger collections, as evidenced by the GOV2 results. Therefore, given this experimental evidence, we decide to set $N = 4$ for use with the basic MRF-SD model.

Now that we have describe why the rationale behind the manual construction of the model, we must see how well it performs compared to the simple MRF-FI model. The results are given in Table 3.6. Results that are statistically significantly better than the MRF-FI model are indicated by a †.

Table 3.6 Test set results for the MRF-SD model. A † indicates a statistically significant improvement over the MRF-FI model

	AP	WSJ	ROBUST04	WT10G	GOV2
MAP	0.2147†	0.3425	0.3096†	0.2053†	0.3325†
GMAP	0.1265	0.2399†	0.2196†	0.1286†	0.2449†
P@10	0.3340	0.5080	0.4566†	0.3245	0.5680†
R-Prec	0.2580†	0.3633	0.3363	0.2374†	0.3716†

The results show that the MRF-SD model is significantly better than the MRF-FI model on every data set except WSJ for mean average precision, which is the primary evaluation metric. The improvements in mean average precision are 3.4% for AP, 5.1% for WSJ, 6.0% for ROBUST04, 10.3% for WT10G, and 11.4% for GOV2. These results indicate very strong, consistent improvements over the bag of words baseline.

Similar results are exhibited for geometric mean average precision. GMAP heavily penalizes queries with a low average precision Therefore, GMAP is often used to measure robustness (Voorhees 2005). As the results show, the MRF-SD model is quite robust and significantly improves GMAP for every data set except AP. We do a deeper analysis of the robustness of the MRF model later in this chapter.

We see that the precision at 10 is improved across most data sets, but is only significant on two of them (ROBUST04 and GOV2). Therefore, it appears as though most of the boost in mean average precision that is achieved from using the MRF-SD model does not come from the very top of the ranked list. Instead, the improvement is likely coming from lower in the ranked list, where the ordered and unordered window features are bringing in more relevant documents and filtering out many of the low ranked, poorly matching documents.

Finally, it is important to recall that training is done to maximize mean average precision. It is likely that more significant improvements could be achieved for the other metrics if the model were trained to optimize them.

6.1.3 Full Dependence

The third basic MRF model is derived from the full dependence model. The model, which is shown in Fig. 3.2 (right), is called the *Full Dependence MRF Model* (MRF-FD model). The model attempts to incorporate dependencies between every subset of query terms and is the most general of the basic models. Here, the number of cliques is exponential in the number of query terms, which restricts the application of this variant to shorter queries. This is not a problem for the MRF-FI and MRF-SD models, which have a linear number of cliques. The model is constructed according

to the following canonical form:

$$
\begin{aligned}
(\text{FI}, T_{QD}, \text{LM}) &: \lambda_{T_D}, \\
(\text{FI}, T_Q, \text{ICF}) &: \lambda_{T_Q}, \\
(\text{FD}, O_{QD}, \text{LM-O-1}) &: \lambda_{O_D}, \\
(\text{FD}, O_Q, \text{ICF-O-1}) &: \lambda_{O_Q}, \\
(\text{FD}, O_{QD}, \text{LM-U-4}) &: \lambda_{U_D}, \\
(\text{FD}, O_Q, \text{ICF-U-4}) &: \lambda_{U_Q}, \\
(\text{FD}, U_{QD}, \text{LM-U-4}) &: \lambda_{U_D}, \\
(\text{FD}, U_Q, \text{ICF-U-4}) &: \lambda_{U_Q}
\end{aligned}
$$

which is similar to the MRF-SD model. However, the models differ in several key ways. First, the MRF-FD model uses the full dependence model type. Second, the MRF-FD model defines two new features for the unordered clique sets (U_{QD} and U_Q). These clique sets are empty in the MRF-SD model. Furthermore, the parameters for all the unordered features are tied together. While this is not required, it simplifies the model.

The resulting joint probability mass function for the model is then given by:

$$
\begin{aligned}
P(Q, D) \propto \exp\Bigg[& \lambda_{T_D} \sum_{(q_i, D) \in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t} \\
& + \lambda_{T_Q} \sum_{q_i \in T_Q} \log \frac{|C|}{cf_{q_i}} \\
& + \lambda_{O_D} \sum_{(q_1, \ldots, q_k, D) \in O_{QD}} \log \frac{tf_{\#1(\{q_i\}),D} + \mu^w \frac{cf_{\#1(\{q_i\})}}{|C|}}{|D| + \mu^w} \\
& + \lambda_{O_Q} \sum_{(q_1, \ldots, q_k) \in O_Q} \log \frac{|C|}{cf_{\#1(\{q_i\})}} \\
& + \lambda_{U_D} \sum_{(q_1, \ldots, q_k, D) \in U_{QD}} \log \frac{tf_{\#\text{uw8}(\{q_i\}),D} + \mu^w \frac{cf_{\#\text{uw8}(\{q_i\})}}{|C|}}{|D| + \mu^w} \\
& + \lambda_{U_Q} \sum_{(q_1, \ldots, q_k) \in U_Q} \log \frac{|C|}{cf_{\#\text{uw8}(\{q_i\})}} \Bigg]
\end{aligned}
\tag{3.23}
$$

Table 3.7 Mean average precision using the MRF-FD model over different combinations of term, ordered, and unordered features

	Term + Ordered				Term + Unordered			
Train\Test	AP	WSJ	WT10g	GOV2	AP	WSJ	WT10g	GOV2
AP	0.185	0.272	0.218	0.267	0.1840	0.267	0.218	0.275
WSJ	0.184	0.273	0.217	0.261	0.1840	0.267	0.218	0.275
WT10G	0.185	0.272	0.218	0.267	0.184	0.267	0.219	0.278
GOV2	0.184	0.271	0.215	0.268	0.184	0.267	0.219	0.278

	Term + Ordered + Unordered			
Train\Test	AP	WSJ	WT10g	GOV2
AP	0.187	0.272	0.223	0.284
WSJ	0.184	0.274	0.220	0.269
WT10G	0.187	0.272	0.223	0.278
GOV2	0.185	0.271	0.220	0.284

and the resulting ranking function is then:

$$
\begin{aligned}
P(D|Q) \overset{\text{rank}}{=} \lambda_{T_D} & \sum_{(q_i, D) \in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t} \\
+ \lambda_{O_D} & \sum_{(q_1,\ldots,q_k, D) \in O_{QD}} \log \frac{tf_{\#1(\{q_i\}),D} + \mu^w \frac{cf_{\#1(\{q_i\})}}{|C|}}{|D| + \mu^w} \\
+ \lambda_{U_D} & \sum_{(q_1,\ldots,q_k, D) \in U_{QD}} \log \frac{tf_{\#\text{uw}4k(\{q_i\}),D} + \mu^w \frac{cf_{\#\text{uw}4k(\{q_i\})}}{|C|}}{|D| + \mu^w}.
\end{aligned}
\tag{3.24}
$$

Now that we have defined the MRF-FD model, we would like to understand what effect the ordered and unordered features have on the model's effectiveness and how well the models learned on one collection generalize to another. In order to measure this, we train on one data set and then use the parameter values found to test on the other data sets. Results for models trained using terms and ordered features, terms and unordered features, and terms, ordered, and unordered features are given in Table 3.7.

For the AP collection, there is very little difference between using ordered and unordered features. However, there is a marginal increase when both ordered and unordered features are used together. The results for the WSJ collection are different. For that collection, the ordered features produce a clear improvement over the unordered features, but there is very little difference between using ordered features

Table 3.8 Test set results for the MRF-FD model. A † indicates a statistically significant improvement over the MRF-FI model and a ‡ indicates statistically significant improvement over the MRF-SD model

	AP	WSJ	ROBUST04	WT10G	GOV2
MAP	0.2128	0.3429†	0.3092†	0.2140†‡	0.3360†
GMAP	0.1257	0.2404†	0.2196†	0.1361†‡	0.2421†
P@10	0.3540	0.5080†	0.45605†	0.3469†‡	0.5720†
R-Prec	0.2543†	0.3694†	0.3394†	0.2417†‡	0.3763†

and the combination of ordered and unordered. The results for the two Web collections, WT10g and GOV2, are similar. In both, unordered features perform better than ordered features, but the combination of both ordered and unordered features led to noticeable improvements in mean average precision.

From these results we can conclude that strict matching via ordered window features is more important for the smaller newswire collections. This may be due to the homogeneous, clean nature of the documents, where an ordered window match is likely to be a high quality match instead of noise. For the Web collections, the opposite is true. Here, the fuzzy unordered window matches provide better evidence. In these less homogeneous, noisy collections, an ordered window match is less likely to be a high quality match and more likely to be a noisy match. Instead, fuzzy matches are appropriate because they deal better with the noise inherent in Web documents.

These results also suggest that parameters trained on any of the data sets generalize well to other data sets. This result is somewhat surprising; we expected parameters trained on newswire (Web) data would generalize better to newswire (Web) test data. However, this is not the case. It appears as though the parameters trained on any reasonable data set will generalize well, which allows one to use a single setting of the parameters across multiple data sets. This may imply that the features used here only capture general aspects of the text and that more domain-specific features may yield further improvements. We return to the issue of parameter generalization later in this chapter.

We conclude our discussion of the MRF-FD model by reporting test set effectiveness results. The results are given in Table 3.8. The improvements over the MRF-FI model are highly consistent, even more so than the improvements we saw with the MRF-SD model. Consistent and significant improvements in mean average precision and geometric mean average precision are observed on every data set except AP. Furthermore, both precision at 10 and R-prec are consistently improved across nearly all of the data sets, as well.

These results indicate that the MRF-FD model is better at improving precision at the top of the ranked list. This suggests that modeling dependencies between non-adjacent query terms, via the use of ordered and unordered features, enhances precision more so than modeling dependencies between adjacent query terms. By using the full dependence model, we impose a more global (i.e., across all query terms) type of proximity constraint on the query terms, whereas the sequential dependence model imposes more of a local (i.e., only between adjacent query terms) proximity

constraint. Hence, the MRF-FD model promotes documents where all of the query terms occur within a close proximity to each other, and the MRF-SD model only promotes documents based on the proximity of pairs of adjacent query terms. The MRF-SD model, therefore, may result in lower quality matches that do not satisfy the global proximity constraints imposed by the MRF-FD model, which may lead to fewer relevant documents returned at the top of the ranked list.

Despite the fact that the MRF-SD model only enforces local proximity constraints, it is only significantly worse than the MRF-FD model on the WT10G data set. The two models are statistically indistinguishable for all other metrics and data sets.

This is an interesting result with practical ramifications. If a system builder had to decide whether to implement the MRF-SD model or the MRF-FD model, they would need to analyze this efficiency/effectiveness tradeoff closely. The results show that, statistically, there is often no difference between the two models. However, as we showed, the MRF-FD model does tend to produce better results across all data sets and metrics. In terms of efficiency, the MRF-SD model requires less computation in order to rank documents, since there are only a linear number of cliques. The MRF-FD model, on the other hand, has an exponential number of cliques. Therefore, the key practical factors to consider are average query length, importance of excellent effectiveness, and computational resources.

Recent advances in inverted indexing technology and query evaluation may be able to significantly improve the efficiency by which both MRF-SD and MRF-FD models can be evaluated. These new techniques, based on impact ordered indexes, pre-compute complicated features and store them directly in the index (Anh and Moffat 2005). Then, rather than computing an exponential number of feature functions per query, the aggregated feature value can be read directly from the index. Of course, applying such an indexing strategy requires a large amount of disk space to store the "feature lists", but could result in very fast query evaluation, especially using recently developed query optimization techniques (Anh and Moffat 2006; Strohman and Croft 2007).

6.2 Evaluation

In this section, we delve deeper into a number of issues related to the three basic MRF models. By analyzing these issues, we help distill a better understanding of the MRF model. Many of the insights described here can be widely applied to other information retrieval models.

6.2.1 Smoothing

All three of the basic models have one or more model hyperparameters that control smoothing. These parameters live outside of the MRF model and must be tuned

separately. Previous research has shown that language modeling effectiveness is often sensitive to the setting of the smoothing parameters (Zhai and Lafferty 2001b). Therefore, it is important to consider how sensitive the effectiveness of the basic models are to the setting of the hyperparameters.

There are two different hyperparameters associated with the basic models. They are μ^t, which controls single term smoothing, and μ^w, which controls both ordered and unordered window smoothing. We choose to smooth single terms different from windows because terms tend to behave differently than windows and have different occurrence statistics. Although not explored here, it is also possible to smooth the ordered window features differently than the unordered windows. However, we feel that the two window types are similar enough that they can be smoothed in the same way.

Since nobody has ever applied smoothing to ordered and unordered windows in this way, it is important to analyze how sensitive effectiveness is with regard to the window smoothing parameters. Figure 3.5 plots the mean average precision surfaces over a wide range of settings for μ^t and μ^w using the MRF-SD model for the AP, WSJ, ROBUST04, and WT10G data set.

The surfaces show that, in general, effectiveness is more sensitive to the setting of the window smoothing parameter (μ^w) than the term smoothing parameter (μ^t). The results suggest that it is important to tune the smoothing parameters, especially the window smoothing parameter. These surfaces also support the decision to smooth windows and terms differently, as it is apparent that setting $\mu^t = \mu^w$ is often far from the optimal setting.

Finally, we note that the AP, WSJ, and WT10G curves are shaped similarly. However, the ROBUST04 surface has a very distinct shape to it. The difference appears to be that it is difficult to "saturate" the window smoothing parameter on the AP, WSJ, and WT10G data sets, but that the window smoothing parameter quickly saturates on the ROBUST04 collection. This result may have to do with the fact that the ROBUST04 query set was specifically chosen to be difficult (for retrieval systems built before 2004). It may be that these queries were "hard" because the models that were applied to them did not take term proximity into account (Buckley 2004; Voorhees 2004). Therefore, when we apply the model to these queries, it may be possible to over smooth the window features, which reduces the effect of term proximity on the ranking function and decreases effectiveness.

6.2.2 Collection Size

In Chap. 1, we described the various paradigm shifts that have occurred in information retrieval. We argued that as collection sizes grew and average document lengths increased, new types of features beyond term frequency and inverse document frequency would become important. We argued that these new types of features that go beyond bags of words would act to filter out all of the noisy matches that were made by chance.

We test this hypothesis by analyzing how much better the MRF-FD model is compared to the MRF-FI model across a range of data set sizes. This data is plotted

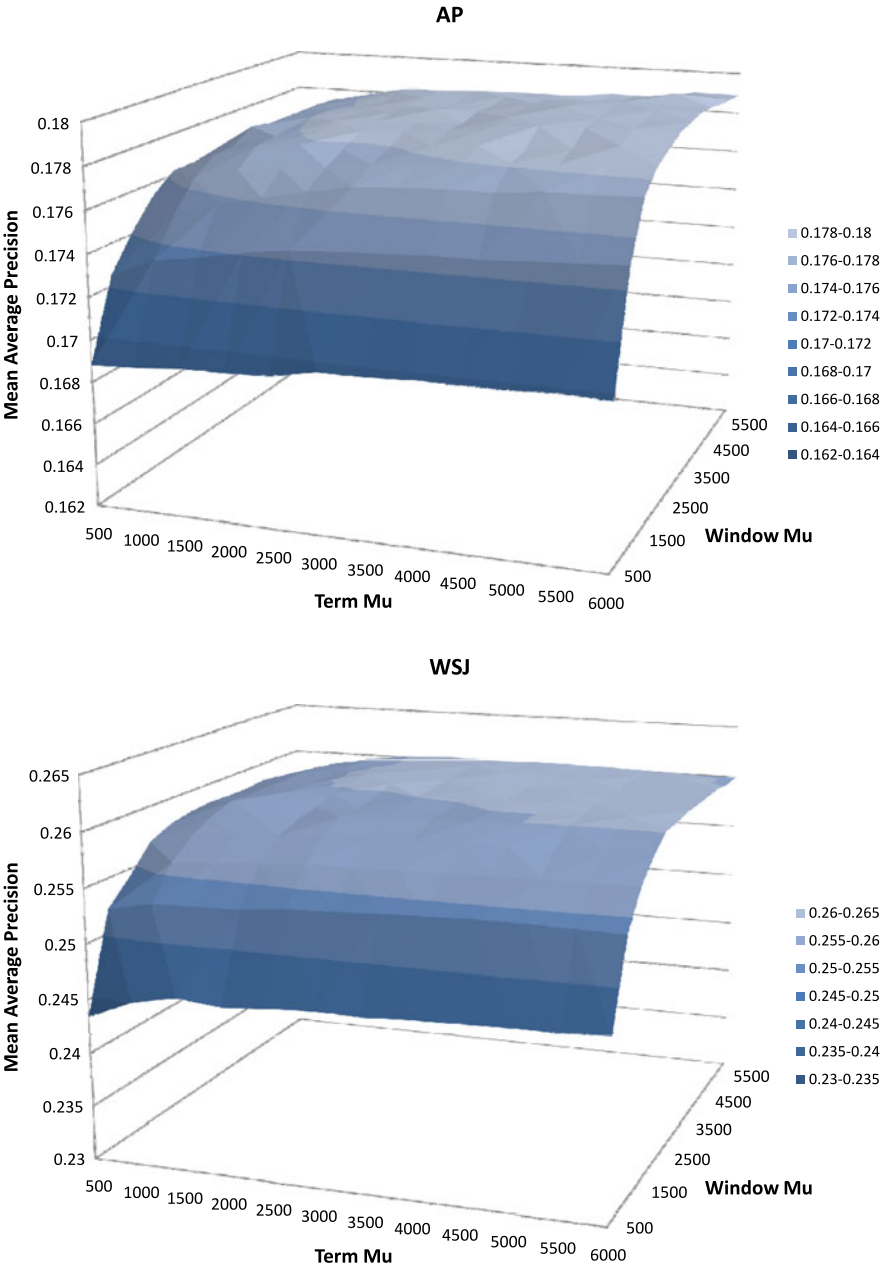

Fig. 3.5 Training set mean average precision as a function of term and window smoothing parameters using the sequential dependence model on the AP, WSJ, ROBUST04, and WT01G data sets (*top to bottom*)

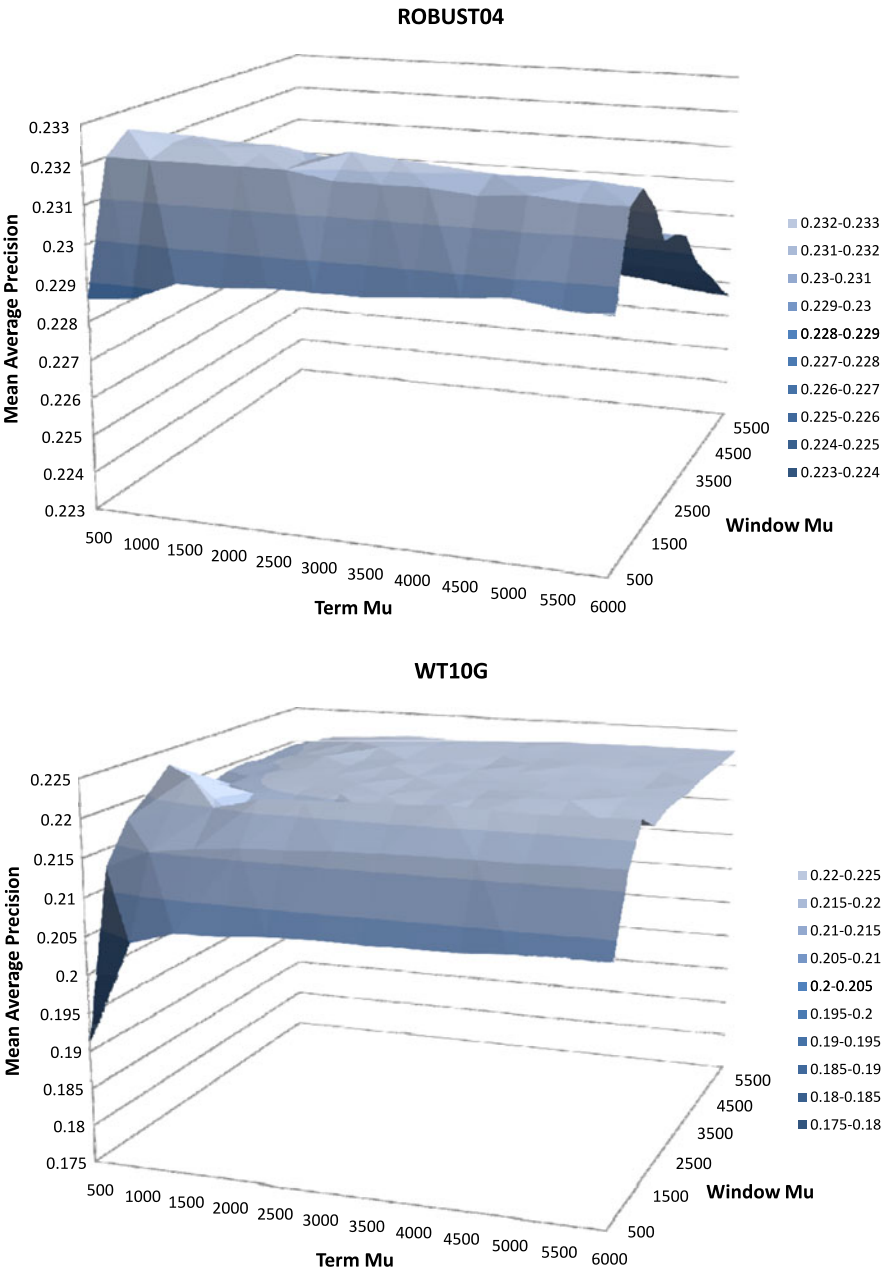

Fig. 3.5 (Continued)

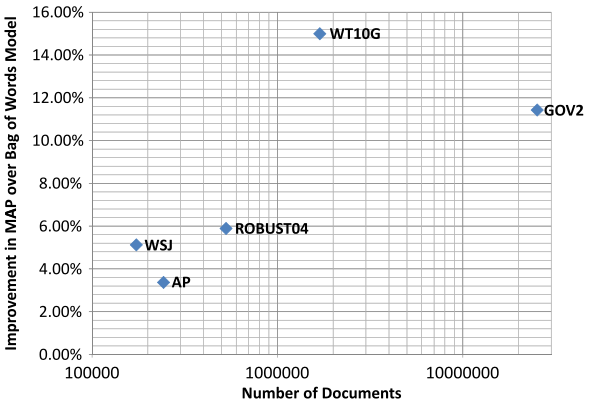

Fig. 3.6 Relationship between the number of documents in a collection and the relative improvement in mean average precision of the MRF-FD model over unigram language modeling (MRF-FI). Note that the x-axis is log scaled

in Fig. 3.6. In the figure, the x-axis represents the number of documents in the collection (log scale) and the y-axis represents the relative improvement in mean average precision of MRF-FD over MRF-FI.

Although there are only five data points, there is clearly an increasing trend in the data, with bigger improvements seen for the larger collections. The trend, of course, is not perfect, but it does help validate the hypothesis that bag of words features begin to fail as collection sizes increase. It will be interesting to see whether this trend continues as larger data sets are made available. Clearly, bag of words models, such as language modeling or BM25, are not well suited for *ad hoc* retrieval against Web-scale collections.

6.2.3 The Role of Features

We have just shown that going beyond the bag of words assumption and making use of term proximity features is highly effective, especially on very large collections. In this section, we take this analysis one step further and investigate the role of various types of features across the data sets. This analysis provides insights into why certain features are more effective than others on a given data set and helps us understand which other types of features may be important in the future.

To aid the analysis, we compute statistics for various information retrieval features. The statistics are computed in the following manner. For every query, we compute the feature of interest for every document in $\mathcal{D}$, the document set of interest. The average feature value is then computed across all of the documents in $\mathcal{D}$. We then use the median of the average values as the primary statistic. We use the median, rather than the average, because many of the feature distributions are highly skewed. This procedure is carried out for the following features:

- **Overlap**—$\frac{|Q \cap D|}{|Q|}$, which is the fraction of query terms that occur in the document. If this value is 1, then every query term occurs in the document. We note that this is similar in spirit to Buckley et al.'s *titlestat* measure (Buckley et al. 2006).

Table 3.9 Median values for various statistics computed across judged relevant (Rel) and non-relevant (Nonrel) documents

	Overlap		Avg. TF		Avg. Dist. (Seq)		Avg. Dist. (Tot)	
	Rel	Nonrel	Rel	Nonrel	Rel	Nonrel	Rel	Nonrel
AP	0.63	0.52	2.6	1.84	212	207	215	210
WSJ	0.67	0.53	2.7	1.93	370	460	387	471
ROBUST04	0.67	0.53	2.9	2.42	430	4933	452	5998
WT10G	0.80	0.66	6.5	5.16	1389	10357	1414	10357
GOV2	0.94	0.76	18.6	18.79	7090	7826	7164	8017

- **Average TF**—$\frac{\sum_{w \in Q} tf_{w,D}}{|Q|}$, which is the average term frequency of the query terms in the document.
- **Average Distance (Sequential)**—The average distance (with respect to term positions) between every pair of query terms that are adjacent to each other. Single term queries are ignored when computing this feature.
- **Average Distance (Total)**—The average distance between *every* pair of query terms. Again, single term queries are ignored when computing this feature.

These features are meant to capture various bag of words features (overlap and average TF), as well as notions of term proximity (average distances).

The medians, as computed using the procedure described above, are given in Table 3.9. Results are given for the AP, WSJ, ROBUST04, WT10G, and GOV2 data sets. The statistics are computed for both the set of judged relevant documents and the set of judged non-relevant documents. By comparing the median values of these features in both sets, we are able to better understand which features discriminate well between relevant and non-relevant documents.

Of course, the judged relevant and judged non-relevant documents are heavily biased because of the pooling procedure used at TREC. However, these statistics still provide valuable insights into the fine line between relevant and non-relevant documents and what types of features are important for data sets with varying characteristics.

We first analyze the overlap feature. As the results show, the overlap is higher in the relevant set than in the non-relevant set. This is to be expected, as relevant documents typically contain most of the query terms. However, there is a noticeable increasing trend in the value as the collection size increases. This suggests that as collections get larger, relevant documents that appear high in the ranked list (i.e., those that would get pooled and judged) will contain most, if not all, of the query terms. This suggests that it might be useful to run the query as a simple conjunctive Boolean query first, and then apply a more complex ranking function to the filtered set of documents.

A similar trend exists for the average term frequency feature, with larger average TF values for larger collections that contain longer documents. This, again, should not be surprising, since collections that contain longer documents will naturally contain more term occurrences. Furthermore, since many of the judgment pools include

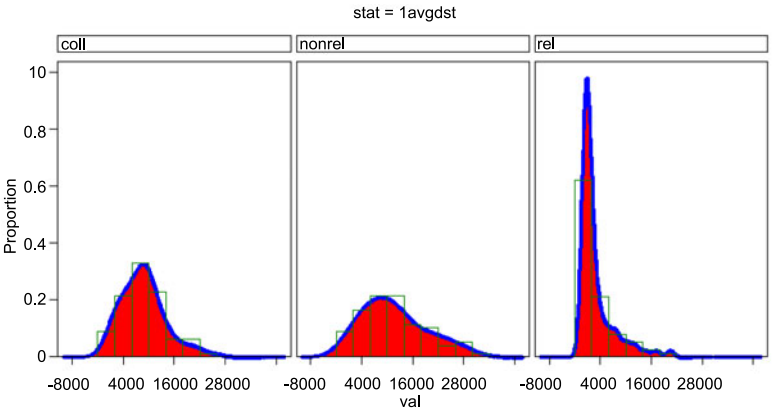

Fig. 3.7 Plot of average distance between adjacent query terms for WT10G data set. *Coll* represents the entire collection, *nonrel* the set of judged non-relevant documents, and *rel* the set of judged relevant documents

a large number of runs that use bag of words models based on *tf.idf* scoring, it is only natural for the results to be biased towards high term frequencies. The gap in TF from the relevant to the non-relevant set is not very large, and therefore average TF alone cannot be used as a very good discriminator. Indeed, it is very interesting to observe that the median average TF of non-relevant documents for GOV2 is larger than the median average TF of relevant documents. This suggests that many of the bag of words models return documents that contain many chance occurrences of the query terms. Since these are not meaningful occurrences of the query terms, the documents were actually non-relevant. This is where the term proximity and other types of features begin to become important, as we will now show.

The two term proximity features show the greatest discriminative potential of any of the features we looked at, especially as collection sizes grow. For the AP and WSJ collections, there is little difference between the term proximity features in the relevant and non-relevant sets. However, for the ROBUST04, WT10G, and GOV2 data sets, there is a noticeable divide between the term proximity characteristics of the relevant and non-relevant document sets. The biggest divide occurs for the WT10G data set, which, not surprisingly, showed the biggest boost in effectiveness when the MRF-SD and MRF-FD models were used. These statistics validate the arguments as to the importance of term proximity features, especially on larger collections. They show that both local proximity, as modeled by the MRF-SD model, as well as global proximity, as modeled by the MRF-FD model, actually model discriminative characteristics of query terms that discriminate between relevant and non-relevant documents.

Finally, in order to give an idea of the distribution of the average distance (sequential) feature, we plot a histogram and smoothed density estimate for the WT10g data set in Fig. 3.7. In addition to the statistics about the judged relevant and non-relevant documents, the same statistics were also computed for the entire set of

documents. The entire set of documents is a much more realistic, less biased model of "not relevant" than the judged non-relevant documents. As the figure shows, the relevant distribution is highly skewed towards small values. The non-relevant distribution is skewed, but not as much as the relevant distribution. The collection distribution does not appear to be as skewed as the other distributions, but it is clear that the average distances compute across the collection are generally much larger than both the judged relevant and non-relevant documents.

6.2.4 Robustness

Next, we investigate the robustness of the MRF-SD and MRF-FD models. Here, we define robustness as the number queries whose effectiveness is improved/hurt (and by how much) as the result of applying these methods. A highly robust model will significantly improve many queries over the baseline and only minimally hurt a few.

In order to evaluate the robustness of the MRF-SD and MRF-FD models, we plot histograms that show how many queries were helped or hurt by a given amount. These plots are given in Fig. 3.8. The bin labels indicate the relative change in mean average precision with regard to the baseline MRF-FI model. We see from the results that the distributions are skewed in the direction of positive improvements. In fact, for most of the data sets, there are very few queries that are hurt by more than 50%. Similarly, many queries are often improved by over 50% on every data set.

These histograms are only useful for aggregate data analysis. However, we would like to know which queries were the most helped and most hurt by these more complex models. In Tables 3.10 and 3.11 we provide the 10 most improved and 10 most hurt queries when using the MRF-SD model on the ROBUST04 and GOV2 data sets, respectively.

The first observation we make about these results is that the most improved queries are often those that have poor MRF-FI average precision (e.g., below 0.1). Of course, since these queries are so poor, it is very easy to achieve large relative improvements. However, some queries, such as *price fixing* (ROBUST04 topic 622), *big dig pork* (GOV2 topic 835), and *spanish civil war support* (GOV2 topic 829) are significantly improved and have "acceptable" MRF-SD average precision values. There are very few cases of queries with large MRF-FI average precisions being significantly hurt when the MRF-SD model is applied to them.

The second observation is that queries consisting of meaningful, common two word phrases, such as *gasoline tax* (ROBUST04 topic 700), *price fixing* (ROBUST04 topic 622), *pol pot* (GOV2 topic 843), and *model railroads* (GOV2 topic 830), are more likely to be improved than two word phrases that are not as common or meaningful, such as *ethnic population* (ROBUST04 topic 651), *tibet protesters* (ROBUST04 topic 612), *eskimo history* (GOV2 topic 837), and *heredity obesity* (GOV2 topic 846). This may be the result of how ordered and unordered feature weights are computed in the MRF-SD model. It may be useful in the future to include notions of lexical cohesiveness in the computation of ordered and unordered phrase features in order to rectify this issue (Vechtomova et al. 2006).

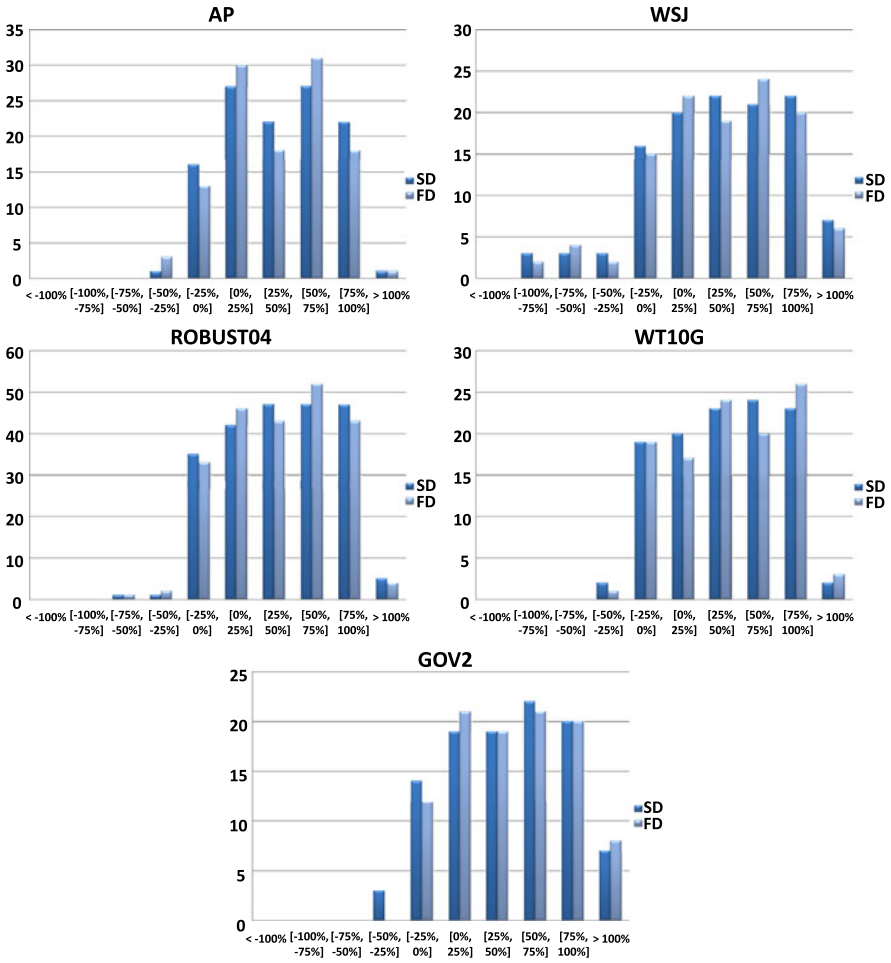

Fig. 3.8 Robustness of MRF-SD and MRF-FD models on the AP, WSJ, ROBUST04, WT10G, and GOV2 test sets. The MRF-FI model is used as the baseline by which the improvements are computed. The evaluation metric used is average precision

Finally, we note that parsing/stopword removal errors may have contributed to the reduction in effectiveness observed for some of the queries. One clear example of such an error is the query *custers stand* (GOV2 topic 822). The original title is *custer's last stand*. However, the query distillation process removes a large set of stopwords, including the term *last*. When the MRF-SD model is applied to the query *custers stand*, the exact phrase feature becomes a very poor feature, since the actual exact phrase features that we want are *custers last* and *last stand*, instead of *custers stand*. Interestingly, the query *doctors borders* (GOV2 topic 806), which is originally *doctors without borders* is the most improved query for the GOV2 data set. A better understanding of stopword removal from within phrases is needed in order to deal with these cases in a more consistent manner.

Table 3.10 The 10 most improved and 10 most hurt test set queries when using the MRF-SD model on the ROBUST04 data set. Effectiveness is measure in terms of average precision

Topic	Query	MRF-FI	MRF-SD	% Change
615	timber exports asia	0.1477	0.0697	−52.81%
685	oscar winner selection	0.2689	0.1740	−35.29%
623	toxic chemical weapon	0.2982	0.2238	−24.95%
651	ethnic population	0.0381	0.0305	−19.95%
659	cruise health safety	0.3216	0.2589	−19.50%
644	exotic animals import	0.1719	0.1392	−19.02%
698	literacy rates africa	0.5278	0.4314	−18.26%
612	tibet protesters	0.4528	0.3733	−17.56%
695	white collar crime sentence	0.2746	0.2316	−15.66%
693	newspapers electronic media	0.3108	0.2680	−13.77%
...				
639	consumer line shopping	0.1353	0.2094	54.77%
629	abortion clinic attack	0.1568	0.2527	61.16%
700	gasoline tax	0.2811	0.4579	62.90%
684	part time benefits	0.0881	0.1482	68.22%
627	russian food crisis	0.0102	0.0175	71.57%
638	wrongful convictions	0.0231	0.0468	102.60%
690	college education advantage	0.0027	0.0062	129.63%
689	family planning aid	0.0224	0.0583	160.27%
666	thatcher resignation impact	0.0078	0.0333	326.92%
622	price fixing	0.0287	0.1354	371.78%

6.2.5 Long Queries

Up until this point, we have only considered queries that were constructed from the title portion of the TREC topics. These queries tend to be very short, high quality queries that contain few, if any, function words. In this section, we examine whether or not the MRF model maintains its effectiveness on long queries. In particular, we are interested in evaluating the model on queries constructed from the description portion of the TREC topic. The description field contains a longer natural language description of the underlying information need and often contains more useful terms. However, it also includes many function words. For this reason, a special stopword list that contains common function words that often occur in TREC description fields was developed. This stopword list is then applied to queries in order to remove most of the noisy query terms, while keeping the important content terms.

In order to evaluate long queries, we define two new basic MRF models designed specifically for long queries. The first, *MRF-FI-L*, is a variant of the MRF-FI model. The *LM* weighting function is replaced with another language modeling weighting

Table 3.11 The 10 most improved and 10 most hurt test set queries when using the MRF-SD model on the GOV2 data set. Effectiveness is measure in terms of average precision

Topic	Query	MRF-FI	MRF-SD	% Change
822	custers stand	0.0815	0.0562	−31.04%
833	iceland government	0.5150	0.3669	−28.76%
847	portugal world war ii	0.2513	0.1859	−26.02%
837	eskimo history	0.0622	0.0486	−21.86%
838	urban suburban coyotes	0.2778	0.2402	−13.53%
846	heredity obesity	0.1734	0.1517	−12.51%
816	usaid assistance galapagos	0.7751	0.7056	−8.97%
850	mississippi river flood	0.1799	0.1663	−7.56%
808	north korean counterfeiting	0.7212	0.6717	−6.86%
845	new jersey tomato	0.3892	0.3635	−6.60%
...				
825	national guard involvement iraq	0.0755	0.1183	56.69%
843	pol pot	0.3158	0.5012	58.71%
849	scalable vector graphics	0.2080	0.4029	93.70%
842	david mccullough	0.0806	0.1799	123.20%
805	identity theft passport	0.0412	0.0961	133.25%
844	segmental duplications	0.0543	0.1486	173.66%
830	model railroads	0.0327	0.0992	203.36%
829	spanish civil war support	0.0637	0.2183	242.70%
835	big dig pork	0.0593	0.2141	261.05%
806	doctors borders	0.0061	0.0750	1129.51%

function based on Jelinek–Mercer smoothing. This is done because previous research has suggested that Jelinek–Mercer smoothing is more effective on longer queries than Dirichlet smoothing (Zhai and Lafferty 2001b). This small change results in the following ranking function:

$$P(D|Q) \overset{\text{rank}}{=} \sum_{(q_i,D)\in T_{QD}} \log\left[\left(1-\delta^t\right)\frac{tf_{q_i,D}}{|D|}+\delta^t\frac{cf_{q_i}}{|C|}\right], \tag{3.25}$$

where δ^t is the term smoothing parameter. In a similar fashion, we modify the MRF-SD model to use Jelinek–Mercer smoothing instead of Dirichlet smoothing. This model has this ranking function:

$$P(D|Q) \overset{\text{rank}}{=} \lambda_{T_D} \sum_{(q_i,D)\in T_{QD}} \log\left[\left(1-\delta^t\right)\frac{tf_{q_i,D}}{|D|}+\delta^t\frac{cf_{q_i}}{|C|}\right]$$

$$+\lambda_{O_D} \sum_{(q_1,q_2,D)\in O_{QD}} \log\left[(1-\delta^w)\frac{tf_{\#1(q_1q_2),D}}{|D|} + \delta^w \frac{cf_{\#1(q_1q_2)}}{|C|}\right]$$

$$+\lambda_{U_D} \sum_{(q_1,q_2,D)\in U_{QD}} \log\left[(1-\delta^w)\frac{tf_{\#uw8(q_1q_2),D}}{|D|}\right.$$

$$\left.+\delta^w \frac{cf_{\#uw8(q_1q_2)}}{|C|}\right], \tag{3.26}$$

where δ^w is the ordered and unordered window smoothing parameter. Both δ^t and δ^w must be in the range [0, 1].

The long query results are given in Table 3.12. These results show that the MRF-SD-L model significantly outperforms the MRF-FI-L model on all data sets. In fact, the improvements here are much larger than those observed for the short queries, which signifies that modeling dependencies between adjacent query terms is even more important for longer queries. One potential reason for this behavior is the fact that longer queries often contain many more spurious terms that, when matched in a bag of words setting, will return many poor documents. Instead, when the adjacency constraint is enforced, the number of these poor matches is reduced.

These results also indicate that the longer queries are much less effective than the shorter version of the queries (see Table 3.6). This poor effectiveness is likely caused by the increased noise in the query. Although many function words are removed from the queries during pre-processing, some make it into the query. Similar results have been observed at TREC, as well (Voorhees 2004, 2005). This brings up the question of whether or not more information, in the form of longer natural language queries, should be expected to return better results than short keyword queries. If the search engine were replaced by a librarian, then it is obvious that the more information that you were to provide, the better the results would ultimately be. However, natural language processing techniques have failed to have a positive effect on retrieval effectiveness, especially for longer queries. Perhaps once natural language techniques are improved, it may be reasonable to expect better effectiveness from longer queries. However, it remains to be seen whether or not users of information retrieval systems will be willing to enter long descriptive queries. Instead, the most likely answer in some technology that lies between short keyword queries and fully descriptive queries. For example, a user interface that allows users

Table 3.12 Test set mean average precision for description-length queries using full and sequential dependence models. All improvements are statistically significant

	MRF-FI-L	MRF-SD-L
AP	0.1778	0.1956 (+10.0%)
WSJ	0.2395	0.2544 (+6.2%)
ROBUST04	0.2910	0.3120 (+7.2%)
WT10G	0.1288	0.1639 (+27.3%)
GOV2	0.2003	0.2412 (+20.4%)

to enter a keyword query and then provides a set of simple options to focus their search results. Such technologies are starting to show up in Web search engines, but whether or not they will enter the mainstream remains yet to be seen.

6.2.6 BM25 Weighting

All of the basic MRFs described so far have used language modeling weighting functions. As described in Chap. 2, the BM25 weighting function has been shown to have effectiveness comparable to language modeling. For this reason, we are interested in examining the effectiveness of MRF models built using BM25 weighting functions. It is straightforward to modify any of the MRF models described thus far to use BM25 weighting. In order to keep things relatively simple and provide for an easy comparison, we choose to modify the MRF-SD model. The resulting MRF-BM25 model is then given by the following canonical form:

$$\begin{gathered}
(\text{FI}, T_{QD}, \text{BM25}) \; : \; \lambda_{T_D}, \\
(\text{FI}, T_Q, \text{IDF}) \; : \; \lambda_{T_Q}, \\
(\text{SD}, O_{QD}, \text{BM25-O-1}) \; : \; \lambda_{O_D}, \\
(\text{SD}, O_Q, \text{IDF-O-1}) \; : \; \lambda_{O_Q}, \\
(\text{SD}, O_{QD}, \text{BM25-U-4}) \; : \; \lambda_{U_D}, \\
(\text{SD}, O_Q, \text{IDF-U-4}) \; : \; \lambda_{U_Q},
\end{gathered}$$

where the weighting functions are defined in Sect. 4.3. The MRF-BM25 model has the same form as the MRF-SD model, but replaces the LM and ICF weighting functions with analogous BM25 and IDF ones. This results in the following ranking function:

$$\begin{aligned}
P(D|Q) \overset{\text{rank}}{=} & \lambda_{T_D} \sum_{(q_i, D) \in T_{QD}} \frac{(k_1^t + 1) tf_{w,D}}{k_1^t \left((1 - b^t) + b^t \frac{|D|}{|D|_{\text{avg}}}\right) + tf_{w,D}} idf(w) \\
& + \lambda_{O_D} \sum_{(q_1, q_2, D) \in O_{QD}} \frac{(k_1^w + 1) tf_{\#1(q_1 q_2),D}}{k_1^w \left((1 - b^w) + b^w \frac{|D|}{|D|_{\text{avg}}}\right) + tf_{\#1(q_1 q_2),D}} \\
& \times idf\left(\#1(q_2 q_2)\right) \\
& + \lambda_{U_D} \sum_{(q_1, q_2, D) \in U_{QD}} \frac{(k_1^w + 1) tf_{\#\text{uw8}(q_1 q_2),D}}{k_1^w \left((1 - b^w) + b^w \frac{|D|}{|D|_{\text{avg}}}\right) + tf_{\#\text{uw8}(q_1 q_2),D}} \\
& \times idf\left(\#\text{uw8}(q_1 q_2)\right)
\end{aligned} \tag{3.27}$$

which has four hyperparameters, as opposed to the two hyperparameters in the MRF-SD model. While the two extra parameters make the model more flexible, to a certain extent, it also makes it more difficult to properly tune.

Table 3.13 Test set results for the MRF-BM25 model. The †, ‡, and ∗ indicate statistically significant improvements over the MRF-FI, BM25 and MRF-SD models, respectively. Recommended term and window hyperparameter values are also provided

	AP	WSJ	ROBUST04	WT10G	GOV2
MAP	0.2210†	0.3512†‡	0.3101†‡	0.2129†‡	0.3476†‡
GMAP	0.1366†∗	0.2471†‡	0.2199†‡	0.1181‡	0.2817†‡∗
P@10	0.3140	0.5140†	0.4525†	0.3388	0.6100†‡∗
R-Prec	0.2666†	0.3698†	0.3366‡	0.2508†‡	0.3834†
(k_1^t, b^t)	(1.75, 0.3)	(1.5, 0.3)	(0.5, 0.3)	(0.5, 0.2)	(1.0, 0.4)
(k_1^w, b^w)	(0.25, 0.1)	(0.25, 0.1)	(0.25, 0.0)	(0.25, 0.0)	(0.25, 0.0)

The results of the experiments using the MRF-BM25 model are shown in Table 3.13. Test set results are given and significance tests are done comparing the retrieval effectiveness against MRF-FI, BM25 (see Eq. 2.11) and MRF-SD. According to the primary evaluation metric, mean average precision, we see that the MRF-BM25 is always significantly better than the MRF-FI model and significantly better than BM25 on all data sets, except AP. Furthermore, the MRF-BM25 model is statistically indistinguishable from the MRF-SD model. These results indicate that the improvement in effectiveness we observed when using the MRF-SD model was not specific to the language modeling weights used. Indeed, as we just showed, similar improvements can be obtained using BM25 weights. Therefore, this general form of model can be used in a "plug 'n play" manner, using any reasonable weighting function.

6.2.7 Comparison to Bigram Model

We now compare the model against another non-bag of words model. The model that we choose to compare against is the bigram language model (see Eq. 2.16), ranked using query likelihood. This model has recently been shown to be one of the most consistently effective non-bag of words models to date (Gao et al. 2004). We compare the effectiveness of the model against the MRF-FI and MRF-SD models. We choose the MRF-SD model since it is a direct generalization of the bigram model and models no additional dependencies, thus making it the most similar model to compare against.

In the experiments, we train the bigram model's smoothing parameters to maximize mean average precision. The test set results are shown in Table 3.14. For each data set, we provide a set of recommended parameter settings. Furthermore, we indicate statistically significant improvements over the MRF-FI model and statistically significant *decreases* in effectiveness versus the MRF-SD model.

The results show that the bigram model is significantly better than the MRF-FI model across all data sets. This result is consistent with previous results (Gao et al. 2004). However, the model is significantly worse than the MRF-SD model on the

Table 3.14 Test set results for the bigram language model. The † indicates a statistically significant improvement over the MRF-FI model and the ↓ indicates a statistically significant *decrease* in effectiveness compared to the MRF-SD model (i.e., MRF-SD > MRF-BM25). Recommended smoothing parameter values are also provided

	AP	WSJ	ROBUST04	WT10G	GOV2
MAP	0.2116†	0.3319†↓	0.3012†↓	0.2165†	0.3371†
GMAP	0.1229	0.2313↓	0.2076†↓	0.1347†	0.2137†↓
P@10	0.3420	0.4880↓	0.4343↓	0.3061	0.5500
R-Prec	0.2537†	0.3561↓	0.3339	0.2499†	0.3744†
μ_1	2750	2250	1000	2750	1250
λ_2	0.995	0.998	0.970	0.950	0.990
λ_3	1.00	1.00	1.00	0.99	1.00

WSJ and ROBUST04 data sets. We note that the bigram model is never significantly better than the MRF-SD model for any metric. This result indicates that while the bigram model can be highly effective, the MRF-SD model is still a better choice, based purely on effectiveness.

Furthermore, we argue that the MRF framework, in general, is always a better choice than the bigram model. Since the MRF model clearly generalizes and supersedes the bigram model, it will always be more flexible and provide more modeling options. Furthermore, the bigram model uses a very rigid set of unigram and bigram features that cannot be changed across tasks. However, the MRF model provides an easy mechanism for including a wide range of arbitrary features. Therefore, there is little reason to choose the bigram model over the MRF model.

One particularly interesting result of the bigram experiments is that the improvement over the MRF-FI model increases as the collection size grows in a similar manner to the MRF-SD model. This result further supports the claim that term dependence and term proximity features are of the utmost importance when collection sizes grow, document lengths increase, and collections become noisier.

6.2.8 Generalization

Finally, we investigate several aspects of how well the MRF model parameters generalize. An underlying goal of parameter selection strategies is to produce a model that generalizes well. A model is said to generalize well if, when trained on one set of data, remains effective on an unseen test set. A model that is capable of achieving excellent effectiveness on a training set but performs poorly on a test set is of minimal value. Therefore, if some parameter selection method results in effectiveness $\hat{m}$, and the optimal effectiveness is m^*, we then compute the following:

$$G = \frac{\hat{m}}{m^*} \tag{3.28}$$

Table 3.15 Intracollection generalization results for mean average precision. Values given are effectiveness ratios

	AP	WSJ	ROBUST04	WT10G	GOV2	Avg.
LM	99.33	99.57	100.0	94.28	99.42	98.52
BM25	99.35	99.67	98.67	98.34	99.79	99.17
F2EXP	100.0	99.97	97.95	95.50	99.66	98.62
MRF-SD	97.93	97.39	99.23	100.0	100.0	98.91

which we define as the *effectiveness ratio*. An ideal model, that generalizes perfectly, would achieve an effectiveness ratio of 1 for every unseen data set. In information retrieval, even a 2–5% change in some measures, such as mean average precision, can be statistically significant, and therefore effectiveness ratios below 0.90 indicate a model's inability to generalize can severely hinder its effectiveness. Most reasonable retrieval models will have an effectiveness ratio greater than 0.95.

We are particularly interested in *intracollection* and *intercollection* generalization, which are two different ways of measuring the generalization properties of a model, which we now describe.

Intracollection generalization deals with how well a model trained on a set of topics from some collection generalizes to another set of topics on that same collection. This is a common setting in TREC evaluations, where collections are often reused from year to year, and systems are typically trained on the topics from the previous year(s).

We ran a number of experiments to test the intracollection properties of various retrieval models. The retrieval models considered are language modeling (LM), BM25, F2EXP (an axiomatic retrieval model that was designed to be less sensitive to parameter estimation (Fang and Zhai 2005)), and the MRF-SD model. In these experiments, parameters are estimated by maximizing mean average precision on the training set. Models are evaluated according to the effectiveness ratio on the test set. The metric used to compute the effectiveness ratio is mean average precision.

The results are given in Table 3.15. The table lists the effectiveness ratios of each model across each data set, as well as the average effectiveness ratio across all data sets. A model with perfect intracollection generalization would have an effectiveness ratio of 100. The results indicate that all of the models do a relatively good job of generalizing, with average effectiveness ratios well above 98%. We note that the F2EXP model tends to generalize better within newswire collections, while the dependence model generalizes better for Web collections. The BM25 model, however, has the best average effectiveness ratio, which indicates its parameters do a particularly good job of capturing collection-dependent characteristics, rather than topic set-specific ones.

The other type of generalization we consider is intercollection generalization. This type of generalization measures how well a model trained on a topic set from one collection generalizes to a different topic set on a different collection. This is a

Table 3.16 Intercollection generalization results. Table includes mean average precision effectiveness ratios across all possible train/test splits using the F2EXP model

Train\Test	AP	WSJ	ROBUST04	WT10G	GOV2	Avg.
AP	–	99.2	94.3	93.0	93.6	95.0
WSJ	99.1	–	97.4	96.4	96.2	97.3
ROBUST04	95.0	97.7	–	97.5	99.6	97.4
WT10G	91.8	92.7	96.5	–	93.4	93.4
GOV2	95.6	98.2	99.3	97.2	–	97.6
Avg.	95.4	96.9	96.9	96.0	95.8	96.2

Table 3.17 Intercollection generalization results. Table includes mean average precision effectiveness ratios across all possible train/test splits using the MRF-SD model

Train\Test	AP	WSJ	ROBUST04	WT10G	GOV2	Avg.
AP	–	100	99.7	98.4	98.9	99.3
WSJ	100	–	99.7	98.4	98.9	99.3
ROBUST04	99.6	99.6	–	99.7	99.3	99.5
WT10G	98.1	98.9	99.8	–	97.0	98.5
GOV2	99.6	99.4	99.7	98.0	–	99.2
Avg.	99.3	99.5	99.7	98.7	98.5	99.1

practical scenario for 'off the shelf' retrieval systems that may be used across a wide range of different collections. It is unlikely that the end users of these systems will be willing or able to provide training data to the system, and therefore the system must be shipped with a very solid set of pre-tuned, highly generalizable parameters.

In order to measure the intercollection generalization, we compute the effectiveness ratio for every possible combination of training/test splits. The results for the F2EXP and MRF-SD models are shown in Tables 3.16 and 3.17, respectively.

As we see from the table, the cross-collection effectiveness ratios for the MRF-SD model are higher for every training/test set pair, with very few exceptions. In fact, on average, the MRF-SD comes within 1% of the optimal setting regardless of which collection is used for training, whereas the F2EXP model only comes within 4% of the optimal on average. The Dirichlet and BM25 models (not shown) have average effectiveness ratios of 98.9% and 96.9%, respectively. Therefore, the MRF-SD model and Dirichlet models are more robust when it comes to cross-collection generalization and make them good candidates for "out of the box" implementations that require a single parameter setting to work well across a wide range of collections.

As further evidence of the model's generalization properties, Fig. 3.9 illustrates the well-behaved, nearly concave surfaces that arise when mean average precision is plotted over the multinomial parameter simplex of the MRF-SD ranking function for various data sets. Each of the mean average precision surfaces has the same

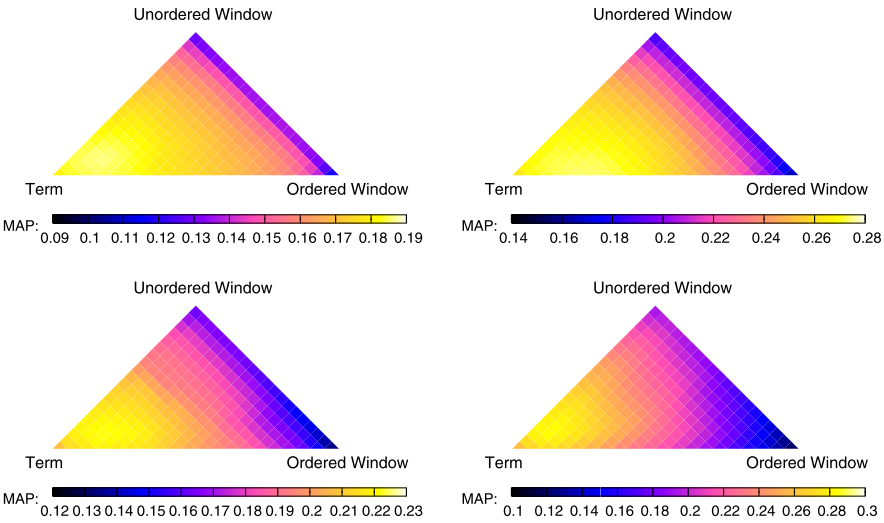

Fig. 3.9 Mean average precision values plotted over MRF-SD parameter simplex for AP, WSJ, WT10g, and GOV2 collections

general form, which indicates that the features capture an inherent property that persists across different types of collections. Although there is no guarantee that such a nicely concave surface will exist for all features and all evaluation metrics, it provides some evidence that the functions we are maximizing over the simplex are not too difficult to optimize using simple algorithms, such as those described in Chap. 6.

6.3 Summary of Results

For completeness, we summarize the main results of the *ad hoc* retrieval experiments in Table 3.18. The table shows test set mean average precision across all data sets for the MRF-FI, MRF-SD, and MRF-FD basic models. It also includes recommended smoothing parameters for each collection, as well as recommended MRF model parameters for each model.

These results show that the hand constructed non-bag of words MRF models (MRF-SD and MRF-FD) consistently outperform the bag of words model (MRF-FI). Although we only show results here that make use of language modeling weighting functions, we note that the results achieved using the MRF-BM25 model, which uses BM25 weighting functions, did consistently and significantly outperform the MRF-FI model in terms of mean average precision across all data sets, thereby satisfying one of the over-arching goals of the model. As we will show in Chap. 6, moving away from hand built MRFs towards automatically constructed ones yields even more significant increases in effectiveness.

Table 3.18 Summary of test set mean average precision for the MRF-FI, MRF-SD, and MRF-FD models across all of the *ad hoc* retrieval data sets. Values in parenthesis denote percentage improvement over MRF-FI model. A † indicates a statistically significant improvement over the MRF-FI model, and a ‡ indicates a statistically significant improvement over the MRF-SD model. Recommended smoothing values are given for each collection, and recommended MRF model parameters are provided for each model

	MRF-FI	MRF-SD	MRF-FD	(μ^t, μ^w)
AP	0.2077	0.2147 (+3.4%)†	0.2128 (+2.5%)	(1750, 5000)
WSJ	0.3258	0.3425 (+4.8%)	0.3429 (+5.2%)†	(2000, 1000)
ROBUST04	0.2920	0.3096 (+6.0%)†	0.3092 (+5.9%)†	(1000, 750)
WT10g	0.1861	0.2053 (+10.3%)†	0.2140 (+15.0%)†‡	(1000, 6000)
GOV2	0.2984	0.3325 (+11.4%)†	0.3360 (+12.6%)†	(1500, 4500)
$(\lambda_{T_D}, \lambda_{O_D}, \lambda_{U_D})$	N/A	(0.85, 0.10, 0.05)	(0.80, 0.10, 0.10)	

7 Web Search

Web search is one of the most popular and widely used information retrieval applications. The goal of a Web search system is to return a set of Web pages that are relevant to a user's query. There are several important differences between *ad hoc* retrieval and Web search. First, not all Web search queries are content-based, or *informational*, searches. For example, a user who enters the query *mcdonalds locations* is not looking for documents that are *about* McDonalds locations. Instead, they are likely searching for the Web page on the McDonald's Web page that lists where their restaurants are located. This type of query, where a user seeks out a specific page that they either know exists or think is very likely to exist, is known as a *navigational* query. Additionally, a user who enters the query *cheap digital cameras* is neither seeking information *about* digital cameras nor seeking a specific page. Instead, the user is likely interested in purchasing a digital camera. Such queries, which are intended to lead to an online transaction, are known as *transactional* queries. A study done by Broder in 2002 reports that approximately 50% of Web queries are informational, 20% navigational, and 30% transactional (Broder 2002).

Since these three query types are so different, they are often evaluated differently. Content-based (informational) Web retrieval was evaluated in the previous section with the experiments on the WT10G and GOV2 data sets. Evaluation of transactional queries is difficult, since it often requires product databases and query click-through logs, which are not currently publicly available for privacy and intellectual property reasons. In this section, we focus on navigational queries, for which there are publicly available TREC data sets.

These data sets were used during the TREC Web Track (1999–2004) and the TREC Terabyte Track (2004–2006). During these tracks, there were several navigational search-related subtasks. Of particular interest to us is the *named page finding* task that was run during the TREC Terabyte Track in 2005 and 2006. We are in-

Fig. 3.10 Example TREC named page finding topics

```
<num> Number: NP1048
<title> us embassy vietnam
</top>

<top>
<num> Number: NP1049
<title> phil english biography
</top>

<top>
<num> Number: NP1050
<title> tenet 9/11 testimony
</top>

<top>
<num> Number: NP1051
<title> hubble timeline
</top>

<top>
<num> Number: NP1052
<title> cdc west nile 2003 statistics by state
</top>
```

terested in this task because of the large data (GOV2) set and the large number of queries available to experiment with (252 from 2005, 181 from 2006).

The named page finding task requires systems to find "bookmarkable" pages that users either know exist or presume are likely to exist (Clarke et al. 2006). One of the main differences between named page finding and *ad hoc* retrieval is that there is typically only one relevant document for every named page finding query. Another key difference is that the primary evaluation metric is mean reciprocal rank, instead of mean average precision[3]. Several example named page finding topics are shown in Fig. 3.10.

In the remainder of this section, we first review previously proposed approaches to Web search (named page finding). We then describe the basic MRF model for Web search. Finally, we evaluate the model on the TREC Terabyte Track named page finding topics.

[3] Average precision is equal to reciprocal rank for queries with only one relevant document, so the two measures will only differ for those topics that have more than one relevant document.

7.1 Previous Models for Web Search

Web data sets are very different than standard *ad hoc* retrieval data sets. They are typically larger and tend to be noisier, because users may publish their own content. Furthermore, Web pages contain HTML markup and can link to other pages. These additional pieces of information play a particularly important role for navigational queries. For this reason, navigational search models often focus on *link structure* and *document structure*.

Link structure refers to the hyperlink structure of the Web. Graph-based algorithms are typically applied to the link structure of the Web in order to find hubs and authoritative pages. Examples of these algorithms include PageRank and HITS (Brin and Page 1998; Kleinberg 1999). These algorithms, along with other Web-specific features such as inlink count, and URL depth have been shown to be useful for improvement the effectiveness of navigational queries (Kraaij et al. 2002).

Methods that make use of document structure often treat certain HTML fields in a special way. For example, a common technique is to weight the importance of text occurring in various fields differently, such as the `title` field or the anchor text pointing to the document (Ogilvie and Callan 2003; Robertson et al. 2004).

Recently, other aspects of Web search, such as user behavior, have been found to be useful, but is beyond the scope of this work (Agichtein et al. 2006).

In this section, we use a bag of words language modeling approach as the baseline. The model, which we refer to as LM-Mixture, makes use of both link structure and document structure, has been shown to be highly effective in the past (Ogilvie and Callan 2003). Given a query, documents are ranked under the model according to:

$$P(D|Q) = \frac{P(D)\prod_{w\in Q}\sum_f P(f|D)P(w|D,f)}{\sum_D P(D)\prod_{w\in Q}\sum_f P(f|D)P(w|D,f)}$$
$$\stackrel{\text{rank}}{=} P(D)\prod_{w\in Q}\sum_f P(f|D)P(w|D,f), \tag{3.29}$$

where $P(w|D,f)$ is the probability of generating term w from field f in document D (i.e., this is a language model built from field f in document D), $P(f|D)$ is a mixing probability, and $P(D)$ is the document's prior probability. It is easy to see that this model is closely related to the standard query likelihood ranking function, except the monolithic document model is replaced with a mixture of field models and a document prior is introduced.

There are other models that attempt to combine evidence from multiple fields, including The BM25F model, which is a field weighted variant of BM25 (Robertson et al. 2004). The model is similar in nature to the mixture of language models approach described here, but field weighting is done differently. Rather than weighting terms after document length normalization is done, as is done in Eq. 3.29, term weights are incorporated before document length normalization. How to properly combine evidence and handle document length normalization in the presence of multiple fields is still an open question (Spärck Jones 2005).

In order to fully specify the model we must describe how to estimate $P(w|D, f)$, $P(f|D)$, and $P(D)$. We begin with the field language model, $P(w|D, f)$. This probability is estimated in a straightforward manner by treating all of the text that appears in field f of document D as a pseudo-document. By doing so, we induce the pseudo-document D_f for field i and the pseudo-collection C_f, which is made up of all of the pseudo-documents constructed from field f. Now, standard language modeling estimation can be applied. We choose to model each field language model as a Dirichlet smoothed language model, which results in the following estimate:

$$P(w|D, f) = \frac{tf_{w,D_f} + \mu_f^t \frac{cf_{w,f}}{|C_f|}}{|D_f| + \mu_f^t}, \tag{3.30}$$

where all of the f subscripts refer to statistics computed in the pseudo-document/pseudo-collection and μ_f^t specifies the smoothing parameter for the field. Since there is a single smoothing parameter per field, accurate estimation may be difficult. Hence, smoothing parameter values are typically chosen to be two times the average length of pseudo-documents of type f. We follow this general rule of thumb in the experiments here.

The mixing probabilities, $P(f|D)$ can either be set to $\frac{|D_f|}{|D|}$ or uniformly. Alternatively, they can be hand or automatically tuned in order to maximize mean reciprocal rank. In this work, we use a set of hand tuned values that have been found to be effective in previous experiments.

7.2 Document Priors

There are many different document priors that can be estimated for navigational Web search tasks. In this section, we describe how to estimate priors based on inlink count and PageRank. The priors are estimated using TREC relevance judgments, although they may also be estimated in a completely unsupervised or semi-supervised setting, given other resources, such as query click logs.

When computing $P(D)$, we really are computing the prior probability that document D is relevant given some external piece of evidence about D, such as the number of links pointing to D or the PageRank of D. Therefore, instead of estimating $P(D)$ directly, we estimate $P(R = 1|\textit{evidence})$, where evidence is some random variable that only depends on the document itself.

7.2.1 Inlink Count

The inlink count of document D is the number of Web pages that point to D. Inlink count is often a good feature to use for navigational queries because the pages that are "bookmarkable" often have a high inlink count associated with them. Therefore, we expect documents with larger numbers of inlinks to have a higher prior probability of relevance.

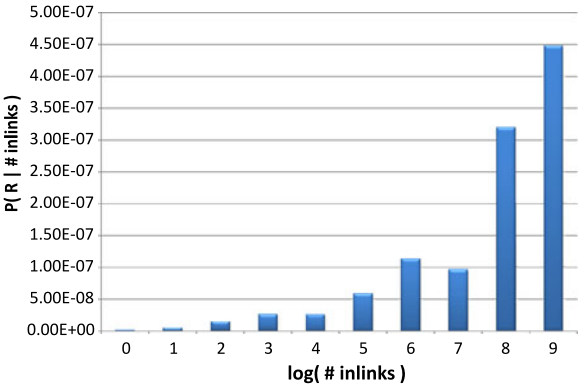

Fig. 3.11 Inlink count prior

Following the framework we described above, we compute the probability of relevance, given the log of the inlink count (simply denoted l). We choose to apply the log function in order to compress the range of values to a more reasonable set. The resulting probability estimate, using Bayes' rule, is:

$$P(R=1|l=X) = \frac{P(l=X|R=1)P(R=1)}{P(l=X)}, \tag{3.31}$$

where $P(l=X|R)$ and $P(R)$ are estimated empirically from TREC relevance judgments. The relevance judgments used are from the TREC 2001 and 2002 Web Track data set. Later, we will apply these priors to the larger TREC 2005–2006 Terabyte Track data set (GOV2). It is unknown how well these probabilities will generalize to this newer, larger data set, but we feel that the estimates should be fairly reasonable.

Figure 3.11 shows the estimated priors across a range of log inlink counts. We see that the prior probability of relevance increases as the number of inlinks increases, as expected.

7.2.2 PageRank

The problem with using inlink count alone is that there is no notion of authority involved. It is very easy for a spammer to create thousands of fake Web pages and have them all point to each other (this is a so-called link farm). This results in a large number of inlinks, but none of the pages are actually authoritative at all. The PageRank algorithm is based on the notion of spreading authority throughout a graph. The basic idea is that if a page has many inlinks from highly authoritative pages, then that page is likely to be authoritative, as well.

The PageRank of document D, denoted $r_{D,t}$, can be computed iteratively. Let t denote the current iteration. Then, the PageRank is computed as follows:

$$r_{D,t+1} = \alpha + (1-\alpha) \sum_{p\,:\,p \to D} \frac{r_{p,t}}{\deg(p)}, \tag{3.32}$$

Table 3.19 URLs in the GOV2 collection with the largest raw PageRank scores. The number of inlinks for each URL is also shown

URL	# Inlinks
http://www.usgs.gov	1,329,036
http://www.ca.gov	471,819
http://www.nih.gov	1,502,324
http://www.epa.gov	1,386,329
http://es.epa.gov/cgi-bin/ncerqamail.pl	1,349,131
http://es.epa.gov/ncer/rfa	1,344,630
http://www.hhs.gov	778,652
http://www.ornl.gov	639,570
http://www.doi.gov	761,456
http://www.medicare.gov	186,948

where $\alpha \in (0, 1]$ affects the amount of "random surfing" done, $p \rightarrow D$ indicates that page p links to document D, and $\deg(p)$ is the number of links out of page p. The values are iteratively updated and renormalized until they converge to the raw PageRank value. There are other ways of computing PageRank, for example, by solving an eigenvector problem, but we use the iterative approach for simplicity.

The raw PageRank is a value between 0 and 1. We assume that the true authoritativeness, or importance, of Web pages is Zipfian in nature. That is, there are a small number of highly authoritative pages, a larger number of less authoritative pages, all the way down to a very large number of non-authoritative pages. Therefore, in order to impose such a distribution, we sort the documents by their raw PageRank scores and then geometrically bin the documents into 11 bins. This idea was inspired by Anh and Moffat's work on document-centric impact weighting (Anh and Moffat 2005). This results in each document being assigned a binned PageRank value between 0 and 10. We use these binned PageRank values in order to estimate the document prior.

The PageRank prior is computed in a similar fashion to the inlink count prior, as follows:

$$P(R = 1|PR = X) = \frac{P(PR = X|R = 1)P(R = 1)}{P(PR = X)}, \tag{3.33}$$

where PR denotes the binned PageRank value. Table 3.19 shows a list of the Web pages in the GOV2 collection with the highest raw PageRank values and the number of inlinks those pages have. Although many of the pages with the highest PageRank have very many inlinks, this is not always the case. The best example of this is the fact that the Medicare Web page has such a high PageRank, but only has 186,948 inlinks.

Lastly, Fig. 3.12 shows the estimated document priors for each binned PageRank value. The plot has an interesting shape to it. Documents with very low PageRank are given a very low prior probability of relevance. The prior probability dramatically increases as the PageRank reaches 8 and 9. Interestingly, a higher prior is assigned to documents with PageRank 8 than 9 and that a zero probability is assigned

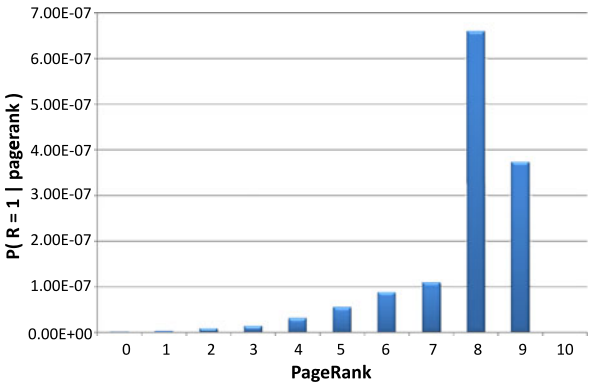

Fig. 3.12 PageRank prior

to documents with PageRank 10. This is very likely the result of data sparseness and the fact that there are so few documents with PageRank 9 or 10 in the relevance judgments.

7.3 MRF Models for Web Search

We now describe one of the many possible ways to use the MRF model for Web search. The mixture of language modeling approach described earlier (see Eq. 3.29) has been shown to be highly effective. Therefore, we wish to use the MRF framework to generalize this model. By doing so, we will be able to model dependencies between query terms, which allows us to make use of phrase and term proximity weighting functions, which were shown to be very valuable for the *ad hoc* retrieval task.

To achieve this, we modify the MRF-SD model by replacing the standard language modeling feature weights (i.e., LM, LM-O-1, and LM-U-4) with analogous mixture of language modeling feature weights. In addition, we add the inlink count and PageRank document priors as feature weights defined over the document clique, as well. These new feature weights, named NP, NP-O-M, NP-U-N, INLINK, and PAGERANK are defined in Table 3.20. This gives rise to the basic MRF for Web search, which we call the *MRF-NP* model, defined by the following canonical form:

$$
\begin{aligned}
(\text{FI}, T_{QD}, \text{NP}) &: \lambda_{T_D}, \\
(\text{FI}, T_Q, \text{ICF}) &: \lambda_{T_Q}, \\
(\text{SD}, O_{QD}, \text{NP-O-1}) &: \lambda_{O_D}, \\
(\text{SD}, O_Q, \text{ICF-O-1}) &: \lambda_{O_Q}, \\
(\text{SD}, O_{QD}, \text{NP-U-4}) &: \lambda_{U_D}, \\
(\text{SD}, O_Q, \text{ICF-U-4}) &: \lambda_{U_Q},
\end{aligned}
$$

Table 3.20 Summary of named page finding weighting functions. The NP, NP-O-M, and NP-U-N weighting functions are based on mixtures of field language models, and INLINK and PAGERANK are based on document priors. The α_f values correspond to the mixing probabilities $P(f|D)$. Term and window smoothing parameters are denoted by μ^t_f and μ^w_f, respectively

NP

$$f_{\text{NP},T}(q_i, D) = \log\left[\sum_f \alpha_f \frac{tf_{q_i,D_f} + \mu^t_f \frac{cf_{q_i,f}}{|C_f|}}{|D_f| + \mu^t_f}\right]$$

NP-O-M

$$f_{\text{NP},O,M}(\{q_i\}, D) = \log\left[\sum_f \alpha_f \frac{tf_{\#M(\{q_i\}),D_f} + \mu^w_f \frac{cf_{\#M(\{q_i\}),f}}{|C_f|}}{|D_f| + \mu^w_f}\right]$$

NP-U-N

$$f_{\text{NP},U,N}(\{q_i\}, D) = \log\left[\sum_f \alpha_f \frac{tf_{\#\text{uw}Nk(\{q_i\}),D_f} + \mu^w_f \frac{cf_{\#\text{uw}Nk(\{q_i\}),f}}{|C_f|}}{|D_f| + \mu^w_f}\right]$$

INLINK

$$f_{\text{INLINK}}(D) = \log P(R = 1|l = l(D))$$

PAGERANK

$$f_{PR}(D) = \log P(R = 1|PR = PR(D))$$

$$(\text{FI},\ D, \text{INLINK})\ :\ \lambda_{IN},$$

$$(\text{FI},\ D, \text{PAGERANK})\ :\ \lambda_{PR}.$$

Using this canonical form, the MRF-NP ranking function is given by:

$$\begin{aligned}
&P(D|Q)\\
&\overset{\text{rank}}{=} \lambda_{T_D} \sum_{(q_i,D)\in T_{QD}} \log\left[\sum_f \alpha_f \frac{tf_{q_i,D_f} + \mu^t_f \frac{cf_{q_i,f}}{|C_f|}}{|D_f| + \mu^t_f}\right]\\
&\quad + \lambda_{O_D} \sum_{(q_1,q_2,D)\in O_{QD}} \log\left[\sum_f \alpha_f \frac{tf_{\#1(q_1q_2),D_f} + \mu^w_f \frac{cf_{\#\ 1(q_1q_2),f}}{|C_f|}}{|D_f| + \mu^w_f}\right]\\
&\quad + \lambda_{U_D} \sum_{(q_1,q_2,D)\in U_{QD}} \log\left[\sum_f \alpha_f \frac{tf_{\#\text{uw}8(q_1q_2),D_f} + \mu^w_f \frac{cf_{\#\text{uw}8(q_1q_2),f}}{|C_f|}}{|D_f| + \mu^w_f}\right]\\
&\quad + \lambda_{IN} \log P\big(R = 1|l = l(D)\big)\\
&\quad + \lambda_{PR} \log P\big(R = 1|PR = PR(D)\big).
\end{aligned} \tag{3.34}$$

The model provides a mechanism for weighting fields (α_f), single term matches (λ_{T_D}), ordered phrases (λ_{O_D}), unordered phrases (λ_{U_D}), inlink prior (λ_{IN}), and PageRank prior (λ_{PR}). Thus, the model encompasses many of the features that have been shown to be effective for navigational Web search.

There are many alternative ways that this model may have been constructed. For example, rather than mixing the field language models within the NP weighting functions, each field language model could be its own feature weight. This would promote the α_f hyperparameters to full-fledged MRF model parameters, which would allow them to be estimated using the machinery described in Chap. 6. Although we do not attempt to empirically analyze the differences in the formulations, we believe that there would be little, if any, difference in the effectiveness of the two models.

7.4 Results

We now empirically evaluate the retrieval effectiveness of the MRF-NP model. For the experiments, we use the TREC 2005 and 2006 Terabyte Track named page finding queries. These queries are evaluated against the GOV2 collection. Please refer to Appendix A for further information on this data set.

In the experiments, we consider four fields in the mixture models. These fields are `body` (full text of the Web page), `title` (text within HTML elements) `heading` (text within `h1`, `h2`, `h3`, and `h4` elements), and `anchor` (all of the anchor text that points to a page). The collection was stemmed using the Porter stemmer and a standard list of 418 stopwords was applied. The MRF model parameters were estimated by maximizing mean reciprocal rank. No more than 1000 results were returned for each query.

Little research has been done on term dependence and non-bag of words models for named page finding. Hence, for experimental purposes, we use the mixture of language models approach as the baseline. In order to ensure fairness, the LM-Mixture model uses the same fields, mixing parameters (α_f), and smoothing parameters (μ_f^t) as the MRF-NP model. In addition, we also use the inlink count and PageRank document priors with the LM-Mixture model. The two priors are combined into a single prior as described in Kraaij et al. (2002).

The results of the experiments are given in Table 3.21. We report results for mean reciprocal rank (MRR), success at rank 10 (S@10), and not found. See Appendix B for definitions of these metrics.

Table 3.21 Summary of named page finding results

	LM-Mixture			MRF-NP		
	MRR	S@10	Not Found	MRR	S@10	Not Found
TREC 2005	0.414	0.563	0.175	0.441	0.583	0.171
TREC 2006	0.472	0.657	0.133	0.512	0.696	0.138

Table 3.22 Results comparing the mean reciprocal rank of the LM-Mixture and MRF-NP models with and without document priors

LM-Mixture		MRF-NP	
No Prior	Prior	No Prior	Prior
0.463	0.472	0.498	0.512

The results show that the MRF-NP model outperforms the baseline language modeling mixture model in terms of MRR on both data sets. There is 6.5% improvement on the 2005 queries and a 8.5% improvement on the 2006 queries. Both of these results are statistically significant. Furthermore, the S@10 metric is improved on both data sets, as well, indicating that the MRF-NP model pulls relevant documents into the top 10. The last metric, not found, does not change significantly for the two models, which indicates that they are both relatively stable in terms of completely failing to find any relevant documents. Although these numbers are relatively good, there is still considerable room for improvement.

One thing that these results do not reveal, however, is how much of the increase in effectiveness comes from the term proximity features and how much comes from the document priors. In Table 3.22, we report the results of an ablation test that attempts to quantify the importance of each type of feature. The results show that document priors improve effectiveness 1.9% on for the LM-Mixture model and 2.8% for the MRF-NP model. This indicates that the MRF model does a better job at combining the evidence from the document priors than the LM-Mixture model. As for the term proximity features, there is an improvement of 7.6% when no priors are used, and an improvement of 8.5% when priors are used. These results show that the term proximity features account for most of the improvement in effectiveness and that, when used in conjunction with the document priors, there is a small additive effect. Therefore, despite what Google would like you to think about PageRank being the heart of their ranking function[4], it appears as though, in reality, PageRank is far less important than fundamental information retrieval features, such as term proximity.

As with the previous analysis, we are interested in developing a better understanding of the types of queries the MRF-based model excels at, and those it fails at. Table 3.23 lists the 10 most helped and 10 most hurt queries from the 2006 data set. These examples do not show any clear trends as to which types of queries are likely to be helped or hurt by the model. In the future, it may be valuable to do an analysis to determine if the most helped queries can be automatically detected using more sophisticated techniques, such as the so-called notion of concept density (Diaz and Metzler 2006).

Finally, we examine the robustness of the MRF-NP method. The results are given in Fig. 3.13. These results are similar to the *ad hoc* retrieval results, with many queries experiencing a large increase in reciprocal rank, and a small number expe-

[4] http://www.google.com/technology/.

Table 3.23 The 10 most improved and 10 most hurt queries on the TREC 2006 Terabyte Track named page finding data set. Effectiveness is measured in terms of reciprocal rank

Topic	Query	LM-Mixture	MRF-NP	% Change
980	medline search	0.00	0.00	−100.0%
962	us iraq rebuilding accomplishments	0.01	<0.00	−80.0%
1038	USPTO Guide and manuals	0.33	0.09	−72.7%
922	marine mammal gray whale	0.33	0.11	−66.6%
906	kickstart deviceprobe	0.33	0.17	−50.00%
943	american folklife center homepage	1.00	0.50	−50.0%
1033	Coastal & Marine Geology InfoBank	0.06	0.04	−33.3%
916	Texas Department of Banking, Agency Philosophy	0.02	0.01	−31.5%
1005	bay trail map	0.20	0.14	−28.6%
1006	olympic games salt lake city new jobs	0.01	0.01	−26.6%
...				
1041	CDC homepage	<0.00	0.01	204.2%
936	patent DRAM cell constructions	0.14	0.50	250.0%
939	Sun Earth student section	0.14	0.50	250.0%
951	us embassy vienna	0.14	0.50	250.0%
984	informal personal caregiver employment	0.13	0.50	300.0%
1030	Space Shuttle Mission #75	0.03	0.14	442.9%
912	Tips for Mobile Homes Residents in Wisconsin	0.03	0.25	650.0%
903	reasons to reduce waste	0.13	1.00	700.0%
1008	land use bill december 2003	0.02	0.17	816.7%
1027	1997 Surface Flows to Nevada from Canadian Province of Origin, by Truck	0.10	1.00	900.0%

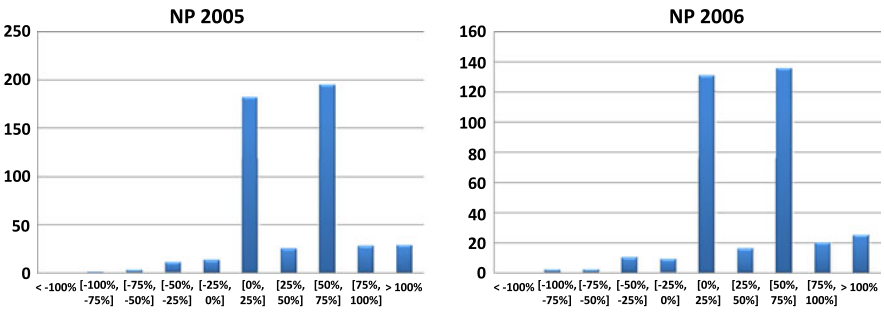

Fig. 3.13 Robustness of the MRF-NP models for the 2005 and 2006 Terabyte Track named page finding data sets. The LM-Mixture model is used as the baseline by which the improvements were computed. The evaluation metric used is reciprocal rank

riencing less significant decreases. The reason why there is a large peak in the [0%, 25%] bin is because the effectiveness of a large number of queries do not change at all. The large peak around [50%, 75%] is likely the result of how the reciprocal rank is computed. If a relevant document moves up in the ranked list by a very small number of positions, then, depending on where in the ranked list it original appeared, the increase in reciprocal rank is likely to fall into this range.

Chapter 4
Feature-Based Query Expansion

1 Overview

Users of information retrieval systems are required to express complex information needs in terms of Boolean expressions, a short list of keywords, a sentence, a question, or possibly a longer narrative. A great deal of information is lost during the process of translating from the information need to the actual query. For this reason, there has been a strong interest in query expansion techniques. Such techniques are used to augment the original query to produce a representation that better reflects the underlying information need.

Query expansion techniques have been well studied for various models in the past and have shown to significantly improve effectiveness in both the relevance feedback and pseudo-relevance feedback setting (Lavrenko and Croft 2001; Rocchio 1971; Xu and Croft 2000; Zhai and Lafferty 2001a).

Until now, the MRF-based models have been solely used for ranking documents in response to a given query. In this chapter, we show how these models can be extended and used for query expansion using a technique that we call *latent concept expansion* (LCE). The approach has three primary benefits.

First, LCE provides a mechanism for combining term dependence with query expansion. Previous query expansion techniques are based on bag of words models. Therefore, by performing query expansion using the MRF model, we are able to study the dynamics between term dependence and query expansion.

Next, query expansion techniques in the past have implicitly only made use of term occurrence features. By using more powerful feature sets, such as those we have described earlier, it is possible to produce better expansion terms that discriminate between relevant and non-relevant documents better.

Finally, the approach described here seamlessly provides a mechanism for generating both single and multi-term concepts. Most previous techniques, by default, generate terms independently. There have been several approaches that make use of generalized concepts, however such approaches were somewhat heuristic and done outside of the model (Papka and Allan 1997; Xu and Croft 2000). The approach is both formally motivated and a natural extension of the underlying model.

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8_4, © Springer-Verlag Berlin Heidelberg 2011

Before describing the details of LCE and formally evaluating it, we review related work in the area of query expansion.

2 Related Work

One of the classic and most widely used approaches to query expansion is the Rocchio algorithm (Rocchio 1971). Rocchio's approach, which was developed within the vector space model, re-weights the original query vector by moving the weights toward the set of relevant or pseudo-relevant documents and away from the non-relevant documents. Unfortunately, it is not possible to formally apply Rocchio's approach to a statistical retrieval model, such as language modeling for information retrieval.

A number of formalized query expansion techniques have been developed for the language modeling framework, including Zhai and Lafferty's model-based feedback and Lavrenko and Croft's relevance models (Lavrenko and Croft 2001; Zhai and Lafferty 2001a). Both approaches attempt to use pseudo-relevant or relevant documents to estimate a better query model.

Model-based feedback finds the model that best describes the relevant documents while taking a background (noise) model into consideration. This separates the content model from the background model. The content model is then interpolated with the original query model to form the expanded query.

The other technique, relevance models, is more closely related to the approach described here. Therefore, we go into the details of the model. Much like model-based feedback, relevance models estimate an improved query model. The only difference between the two approaches is that relevance models do not explicitly model the relevant or pseudo-relevant documents. Instead, they model a more generalized notion of relevance, as we now show.

Given a query Q, a relevance model is a multinomial distribution over the vocabulary, $P(\cdot|Q)$, that encodes the likelihood of each term given the query as evidence. It is computed as

$$\begin{aligned} P(w|Q) &= \int_D P(w|D)P(D|Q) \\ &\approx \frac{\sum_{D\in\mathcal{R}_Q} P(w|D)P(Q|D)P(D)}{\sum_w \sum_{D\in\mathcal{R}_Q} P(w|D)P(Q|D)P(D)}, \end{aligned} \tag{4.1}$$

where $\mathcal{R}_Q$ is the set of documents that are relevant or pseudo-relevant to query Q. In the pseudo-relevant case, these are the top ranked documents for query Q. Furthermore, it is assumed that $P(D)$ is uniform over this set. These mild assumptions make computing the Bayesian posterior more practical.

After the model is estimated, documents are ranked by clipping the relevance model by choosing the k most likely terms from $P(\cdot|Q)$. This clipped distribution is then interpolated with the original, maximum likelihood query model (Metzler et al.

2004b). This can be thought of as expanding the original query by k weighted terms. Throughout the remainder of this work, we refer to this instantiation of relevance models as RM3.

There has been relatively little work done in the area of query expansion in the context of dependence models (Harper and van Rijsbergen 1978). However, there have been several attempts to expand using multi-term concepts. Xu and Croft's local context analysis (LCA) method combined passage-level retrieval with concept expansion, where concepts were single terms and phrases (Xu and Croft 2000). Expansion concepts were chosen and weighted using a metric based on co-occurrence statistics. However, it is not clear based on the analysis done how much the phrases helped over the single terms alone.

Papka and Allan investigate using relevance feedback to perform multi-term concept expansion for document routing (Papka and Allan 1997). The concepts used in their work are more general than those used in LCA, and include Indri query language structures, such as `#UW50`(white house), which corresponds to the concept "the terms white and house occur, in any order, within 50 terms of each other". Results showed that combining single term and large window multi-term concepts significantly improved effectiveness. However, it is unclear whether the same approach is also effective for *ad hoc* retrieval, due to the differences in the tasks.

3 Basic Latent Concept Expansion

We now describe how the MRF framework can be used to generate single and multi-term concepts that are topically related to some original query. As we will show, the concepts generated using the technique can be used for query expansion or other tasks, such as suggesting alternative query formulations.

We assume that when a user formulates their original query, they have some set of concepts in mind, but are only able to express a small number of them in the form of a query. We treat the concepts that the user has in mind, but did not explicitly express in the query, as latent concepts. These latent concepts can consist of a single term, multiple terms, or some combination of the two. It is our goal to recover these latent concepts given some original query.

This can be accomplished within the framework by first expanding the original MRF graph G to include the type of concept we are interested in generating. We call this expanded graph H. In Fig. 4.1, the middle graph provides an example of how to construct an expanded graph that can generate single term concepts. Similarly, the graph on the bottom illustrates an expanded graph that generates two term concepts. Although these two examples make use of the sequential dependence model (i.e., dependencies between adjacent query terms), it is important to note that both the original query and the expansion concepts can use any dependence structure.

After H is constructed, we compute $P_{H,\Lambda}(E|Q)$, a probability distribution over latent concepts, according to

$$P_{H,\Lambda}(E|Q) = \frac{\sum_{D \in \mathcal{R}} P_{H,\Lambda}(Q, E, D)}{\sum_{D \in \mathcal{R}} \sum_{E} P_{H,\Lambda}(Q, E, D)}, \tag{4.2}$$

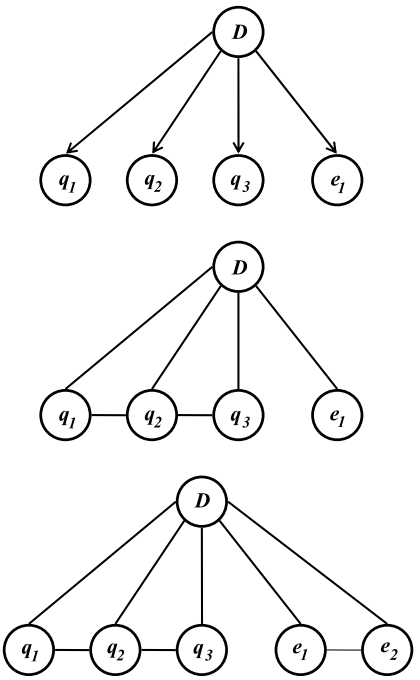

Fig. 4.1 Graphical model representations of relevance modeling (*top*), latent concept expansion using single term concepts (*middle*), and latent concept expansion using two term concepts (*bottom*) for a three term query

where $\mathcal{R}$ is the universe of all possible documents and E is some latent concept that may consist of one or more terms. Since it is not practical to compute this summation, we must approximate it. We notice that $P_{H,\Lambda}(Q, E, D)$ is likely to be peaked around those documents D that are highly ranked according to query Q. Therefore, we approximate $P_{H,\Lambda}(E|Q)$ by only summing over a small subset of relevant or pseudo-relevant documents for query Q. This is computed as follows:

$$P_{H,\Lambda}(E|Q) \approx \frac{\sum_{D \in \mathcal{R}_Q} P_{H,\Lambda}(Q, E, D)}{\sum_{D \in \mathcal{R}_Q} \sum_E P_{H,\Lambda}(Q, E, D)}, \tag{4.3}$$

where $\mathcal{R}_Q$ is a set of relevant or pseudo-relevant documents for query Q and all clique sets are constructed using H.

As an example, suppose that we were to perform LCE on a MRF-SD model and expand by single term concepts. Then, the graph H would be like the middle one in Fig. 4.1. Under this setting, expansion concepts would be generated in the following way:

$$P(e|Q) \propto \sum_{D \in \mathcal{R}_Q} \exp\left[\lambda_{T_D} \sum_{(q_i, D) \in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D| + \mu^t}\right.$$

$$+\lambda_{O_D} \sum_{(q_1,q_2,D)\in O_{QD}} \log \frac{tf_{\#1(q_1q_2),D} + \mu^w \frac{cf_{\#1(q_1q_2)}}{|C|}}{|D|+\mu^w}$$

$$+\lambda_{U_D} \sum_{(q_1,q_2,D)\in U_{QD}} \log \frac{tf_{\#uw8(q_1q_2),D} + \mu^w \frac{cf_{\#uw8(q_1q_2)}}{|C|}}{|D|+\mu^w}$$

$$+\lambda'_{T_D} \log \frac{tf_{e,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D|+\mu^t}$$

$$+\lambda'_{T_Q} \log \frac{|C|}{cf_e}\Bigg]. \tag{4.4}$$

As we see, the likelihood contribution for each document in $\mathcal{R}_Q$ is a combination of the original query's score for the document (first 3 components, refer to Eq. 3.22), concept E's score for the document (fourth component), and E's document-independent score (fifth component). Therefore, this equation can be interpreted as measuring how well Q and E account for the top ranked documents and the "goodness" of E, independent of the documents. For maximum flexibility, we introduce a new set of parameters for the expansion concept (i.e., λ'_{T_D} and λ'_{T_Q}), which allows us to weight the expansion features differently than those in the original query.

For the sake of completeness, we show a more complex example, where the original query uses the MRF-FD model and the expansion concept, which consists of two terms, uses the MRF-SD model. Under this model, concept probabilities are computed according to

$$P(e_1,e_2|Q) \propto \sum_{D\in\mathcal{R}_Q} \exp\Bigg[\lambda_{T_D} \sum_{(q_i,D)\in T_{QD}} \log \frac{tf_{q_i,D} + \mu^t \frac{cf_{q_i}}{|C|}}{|D|+\mu^t}$$

$$+\lambda_{O_D} \sum_{(q_1,\ldots,q_k,D)\in O_{QD}} \log \frac{tf_{\#1(\{q_i\}),D} + \mu^w \frac{cf_{\#1(\{q_i\})}}{|C|}}{|D|+\mu^w}$$

$$+\lambda_{U_D} \sum_{(q_1,\ldots,q_k,D)\in U_{QD}} \log \frac{tf_{\#uw8(\{q_i\}),D} + \mu^w \frac{cf_{\#uw8(\{q_i\})}}{|C|}}{|D|+\mu^w}$$

$$+\lambda'_{T_D} \sum_{i=1}^{2} \log \frac{tf_{e_i,D} + \mu^t \frac{cf_{e_i}}{|C|}}{|D|+\mu^t}$$

$$+\lambda'_{O_D} \log \frac{tf_{\#1(e_1e_2),D} + \mu^w \frac{cf_{\#1(e_1e_2)}}{|C|}}{|D|+\mu^w}$$

$$+\lambda'_{U_D}\log\frac{tf_{\#\text{uw8}(e_1e_2),D}+\mu^w\frac{cf_{\#\text{uw8}(e_1e_2)}}{|C|}}{|D|+\mu^w}$$
$$+\lambda'_{T_Q}\sum_{i=1}^{2}\log\frac{|C|}{cf_{e_i}}+\lambda'_{O_Q}\log\frac{|C|}{cf_{\#1(e_1e_2)}}$$
$$+\lambda'_{U_Q}\log\frac{|C|}{cf_{\#\text{uw8}(e_1e_2)}}\Bigg], \tag{4.5}$$

where the first three components compute the MRF-FD score of the original query, the next three components compute the MRF-SD score of the expansion concept, and the last three components compute an IDF-based "goodness" score of the concept.

3.1 Query Expansion

To use this framework for query expansion, we first choose an expansion graph H that encodes the latent concept structure we are interested in expanding the query using. We then select the k latent concepts with the highest likelihood given by Eq. 4.3. A new graph G' is constructed by augmenting the original graph G with the k expansion concepts $e_1, \ldots, e_k$. Finally, documents are ranked according to $P_{G',\Lambda}(D|Q, e_1, \ldots, e_k)$.

3.2 Comparison to Relevance Models

Inspecting Eqs. 4.1 and 4.3 reveals the close connection that exists between LCE and relevance models. Both models essentially compute the likelihood of a term (or concept) in the same manner. It is easy to see that just as the MRF model can be viewed as a generalization of language modeling, so too can LCE be viewed as a generalization of relevance models. In fact, by setting $\lambda_{T_D} = \lambda'_{T_D} = 1$ and all other parameters to 0 in Eq. 4.4, we derive the relevance model formula.

There are important differences between MRFs/LCE and unigram language models/relevance models. See Fig. 4.1 for graphical model representations of both models. Unigram language models and relevance models are based on the multinomial distribution. This distributional assumption locks the model into the bag of words representation and the implicit use of term occurrence features. However, the distribution underlying the MRF model allows us to move beyond both of these assumptions, by modeling both dependencies between query terms and allowing arbitrary features to be explicitly used.

Moving beyond the simplistic bag of words assumption in this way results in a general, robust model and, as we show in the next section, translates into significant improvements in retrieval effectiveness.

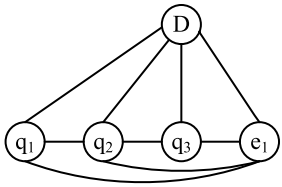

Fig. 4.2 Graphical model representation of generalized latent concept expansion for a three term query

4 Generalized Latent Concept Expansion

The basic version of LCE assumes that query terms and latent concepts are conditionally independent given a document. It is likely that this independence assumption can degrade retrieval effectiveness, because it is important to expand the query with concepts that are actually related to the original query terms. Therefore, we extend the model structure of LCE, by adding edges between each query term node and the latent concept node. We refer to this generalized model as generalized LCE, or LCE-GE. The graphical model representation of the model is shown in Fig. 4.2. With the newly added edges, LCE-GE can explicitly model the dependencies between query terms and latent concepts.

Following the original LCE approach, we parameterize the MRF model based on clique sets to provide more flexibility in encoding useful features over cliques in the graph while keeping the number of features and parameters reasonable. Cliques in the same clique set can share feature functions and parameters, which significantly reduce the parameter space. Meanwhile, we can tune the parameters on the level of clique sets, which is more effective and efficient than tuning on the level of cliques for information retrieval. We expand upon the clique sets used by LCE and ultimately make use of seven clique sets. The clique sets used by generalized LCE are summarized in Table 4.1.

The first three clique sets consist of cliques containing query term nodes (or the latent concept node) and the document node. Features over these cliques encode how well the terms in the clique describe the document. The next three clique sets, which are novel for LCE-GE and cannot be captured by the "basic" LCE, consist of cliques containing query term nodes, the latent concept node, and the document node. Features over these cliques encode the co-occurrence relationship between query terms and latent concepts in the relevant or pseudo-relevant documents. Finally, there is the clique only contains the document node. Features over this node can be used as a type of document prior, encoding document-centric properties.

After tying the parameters based on the clique sets, the joint distribution is defined as

$$P(Q,E,D) = \frac{1}{Z}\exp\Bigg\{\lambda_{T_D}\sum_{c\in T_D} f_{T_D}(c) + \lambda_{O_D}\sum_{c\in O_D} f_{O_D}(c) + \lambda_{U_D}\sum_{c\in U_D} f_{U_D}(c) + \lambda_{T_{QE}}\sum_{c\in T_{QE}} f_{T_{QE}}(c)$$

$$+\lambda_{O_{QE}} \sum_{c \in O_{QE}} f_{O_{QE}}(c) + \lambda_{U_{QE}} \sum_{c \in U_{QE}} f_{U_{QE}}(c)$$
$$+\lambda_D f_D(D) \Big\}. \tag{4.6}$$

4.1 Features

To utilize the model, we must define the feature functions $f(c)$. The correct choice of features depends largely on the retrieval task and the evaluation metric. It is unlikely there is a single, universally applicable set of features. Since the goal here is not to find optimal features, we use a simple, fixed set of features that have shown to be effective in previous work (Metzler and Croft 2005, 2007). Table 4.2 lists the features used with the generalized LCE model. These features attempt to capture various types of term occurrence and term proximity information.

4.2 Comparison with Previous Models

It is important to note that relevance-based language models, LCE, and generalized version of LCE are all closely connected. The core difference between the approaches, as is clearly illustrated by Fig. 4.2, is how the joint distribution $P(Q, D, E)$ is defined. Relevance models make the most assumptions (directed

Table 4.1 Summary of generalized LCE clique sets

Name	Description
T_D	set of cliques containing the document node and exactly one query term node or the latent concept node
O_D	set of cliques containing the document node and two or more query term nodes that appear in sequential order within the query
U_D	set of cliques containing the document node and two or more query term nodes that appear in any order within the query
T_{QE}	set of cliques containing the document node, the latent concept node, and exactly one query term node
O_{QE}	set of cliques containing the document node, the latent concept node, and two or more query term nodes that appear in sequential order within the query
U_{QE}	set of cliques containing the document node, the latent concept node, and two or more query term nodes that appear in any order within the query
D	set of cliques containing only the document node

Table 4.2 Feature functions used for generalized latent concept expansion. Here, q is a query term, b is a query bigram, $tf_{w,D}$ is the number of times term w occurs in document D, $tf_{\#1(b),D}$ denotes the number of times the exact phrase b occurs in document D, $tf_{\#uw(b),D}$ is the number of times the terms in b appear ordered or unordered within a window of 8 terms, and $|D|$ is the length of document D. The cf and $|C|$ values are analogously defined on the collection level. Finally, α and β are model hyperparameters that control smoothing for single term and phrase features, respectively

Feature	Value
$f_{T_D}(q, D)$	$\log[(1-\alpha)\frac{tf_{q,D}}{\|D\|} + \alpha\frac{cf_q}{\|C\|}]$
$f_{O_D}(b, D)$	$\log[(1-\beta)\frac{tf_{\#1(b),D}}{\|D\|} + \beta\frac{cf_{\#1(b)}}{\|C\|}]$
$f_{U_D}(b, D)$	$\log[(1-\beta)\frac{tf_{\#uw(b),D}}{\|D\|} + \beta\frac{cf_{\#uw(b)}}{\|C\|}]$
$f_{T_{QE}}(q, E, D)$	$\log[(1-\beta)\frac{tf_{\#uw(q,E),D}}{\|D\|} + \beta\frac{cf_{\#uw(q,E)}}{\|C\|}]$
$f_{O_{QE}}(b, E, D)$	$\log[(1-\beta)\frac{tf_{\#uw(b,E),D}}{\|D\|} + \beta\frac{cf_{\#uw(b,E)}}{\|C\|}]$
$f_{U_{QE}}(b, E, D)$	$\log[(1-\beta)\frac{tf_{\#uw(b,E),D}}{\|D\|} + \beta\frac{cf_{\#uw(b,E)}}{\|C\|}]$
f_D	0

model, conditional independence between q_i and e), while generalized LCE makes the fewest (undirected model, no conditional independence between q_i and e).

Although LCE-GE approach can incorporate all of the dependencies essential for the query expansion task, its model structure may not be the most appropriate for the following reasons.

High Computational Complexity In LCE-GE, the latent concept now explicitly depends on the user query. Since a latent concept could be any term in the document collection with some probability, there are large amount of pairs of query terms and latent concepts that must be evaluated. Moreover, evaluating term proximity features can be time consuming for most retrieval systems compared to evaluating term occurrence features, especially when the retrieval system uses standard positional inverted indices. As a result, it may be practically infeasible to implement the LCE-GE approach in a large-scale retrieval system. We will provide a more formal analysis of the time complexity of LCE-GE in Sect. 5.4.

Data Sparseness Data sparseness refers to the fact that most query terms only have a few number of occurrences distributed in a relatively long document and hence a very sparse query term-latent concept relationship matrix is generated. This problem is also compounded by the fact that term proximity feature functions are often more sparse than term occurrence feature functions, since term co-occurrences must be within a window size (e.g., 8 terms in LCE-GE).

In the next section, we will introduce a different, implicit way of modeling dependencies between query terms and expansion concepts that can be computed more efficiently than LCE-GE while maintaining strong retrieval effectiveness.

5 LCE Using Hierarchical MRFs

In this section, we describe the Hierarchical Markov random field model, which can be used in conjunction with LCE to construct a highly effective query expansion approach.

5.1 Data Representation

For the query expansion task, it is important to use a representation that makes full use of the document as evidence. Most previous approaches use unstructured bag of words document representations, and therefore can only explore co-occurrence features at the document level. This type of representation is too coarse for our needs, especially for long documents which are more susceptible to topic drift (Macdonald and Ounis 2007).

Instead, we need a representation that allows us to explore co-occurrence at different spatial scales within the document, such as sentence, paragraph, or passage-level scales. Therefore, we represent documents using a tree structure. Each node in the tree represents a content region within the document, with the root node representing the whole document. Each node is the aggregation of all its children nodes. All leaf nodes are basic content units and form a flat segmentation of the document.

We are interested in partitioning documents into topical segments. Some learning methods have recently been proposed to automatically infer document structure from unstructured documents (Blei et al. 2003b; Ji and Zha 2003). However, such methods may be difficult to implement, computationally expensive, or simply inappropriate for the given task. Therefore, to approximate the effect of representing a document as a tree structure, we represent a document as a *two-layer tree*. The first layer contains a single root node representing the entire document, and the second layer contains nodes that partition the document into fixed length segments (passages). Thus, segments within the document are the basic content units now, instead of the entire document. In principle though, there is nothing preventing the model described in this section from extending beyond two layers or segmenting documents in some other way.

Although the introduced two-layer tree can describe multiple scale information of a document, it is inappropriate to use it as a basic data unit to explore co-occurrence for query expansion. Since every document can have a different number of segments, the structure of the two-layer tree is document specific and thus changes for every document. If the entire two-layer tree is used as a basic data unit to explore co-occurrence, models with different structures are needed for different documents, which would easily lead to overparameterization.

In this work, we are interested in triplets of nodes $\{S_{j-1}, S_j, D\}$ from the two-layer tree. Nodes S_{j-1} and S_j are associated with neighboring sibling nodes in the second layer of the document tree (i.e., they are adjacent segments), while node D is the root node of the tree (i.e., the entire document). Here, we use the positional

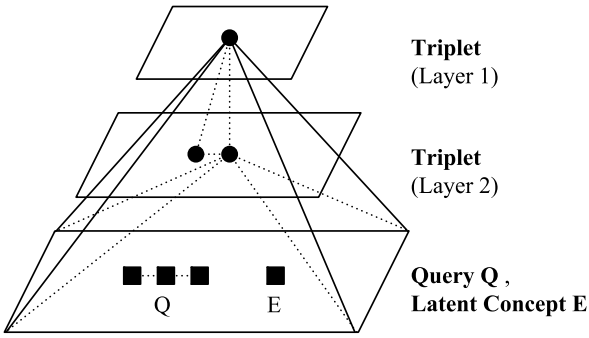

Fig. 4.3 Illustration of LCE-HMRF structure for a three term query. *Circles* represent a triplet of nodes extracted from the document tree. Layer 1 has a single node that is associated with the root node at the first layer of the document tree and represents the whole document. Layer 2 contains two nodes that are associated with neighboring sibling nodes at the second layer of the document tree and represent basic content units (i.e. segments) within the document. *Squares* represent query terms and a latent concept, which are under the same independence assumption adopted by LCE. That is, neighboring query terms are linked, and both query terms and the latent concept are linked to all the nodes of the triplet. Note that there is no link between query terms and the latent concept

information to sequentialize the nodes on the tree from left to right. The triplets extracted from a single document form a set and cover all the nodes of the document tree, but are not a partition, since some nodes appear in multiple triplets. The remainder of this section describes how MRFs can be used to model a joint distribution over a query, a latent concept, and this triplet, and how the distribution can ultimately be used for query expansion.

5.2 Model Description

Given the hierarchical data representation and the triplets extracted from the document tree, a Hierarchical Markov random field (HMRF) model can easily be constructed and used to generate latent concepts. When HMRFs are used in this way (i.e., within the LCE framework to generate latent concepts), we refer to the resulting approach as LCE-HMRF. Figure 4.3 provides a high level illustration of the LCE-HMRF model structure. We assume there are dependencies between nodes that are associated with parent and child nodes in the document tree. Meanwhile, we assume a dependency exists between nodes that are associated with neighboring sibling nodes at the same layer of the document tree. Query terms and the latent concept are under the same independence assumptions adopted by LCE, that is, neighboring query terms are linked, and both query terms and the latent concept are linked to all the nodes of the triplet. Note that there are *no* edges between query terms and the latent concept as in LCE-GE.

We parameterize LCE-HMRF based on clique sets in the same way as LCE-GE. Unlike LCE-GE, LCE-HMRF can exploit different levels of contextual information

Table 4.3 Two additional clique sets used with LCE-HMRF

Name	Description
T_{SQ}	set of cliques containing exactly one query term node, the root node at the first layer of the document tree, and two neighboring nodes at the second layer of the document tree.
T_{SE}	set of cliques containing the latent concept node, the root node at the first layer of the document tree, and two neighboring nodes at the second layer of the document tree.

in a document. We describe two novel clique sets, which provide a mechanism to exploit a document's fine-grained evidence. Features over these cliques encode how well the terms in the clique describe a basic content unit (i.e. a segment) within the document. The two new clique sets are described in Table 4.3. Meanwhile, LCE-HMRF can also exploit a document's coarse-grained evidence, like LCE-GE. That is, the clique sets T_D, O_D, U_D, D that are used for LCE-GE are also used here, where the original document node on the LCE-GE graph is replaced by the root node at the first layer of the document tree. Putting everything together, the joint distribution of LCE-HMRF can be written as

$$\begin{aligned} &P(Q, E, S_{j-1}, S_j, D) \\ &= \frac{1}{Z} \exp\Bigg\{ \underbrace{\lambda_{T_D} \sum_{c \in T_D} f_{T_D}(c) + \lambda_{O_D} \sum_{c \in O_D} f_{O_D}(c) + \lambda_{U_D} \sum_{c \in U_D} f_{U_D}(c)}_{F_D(Q)+F_D(E)\text{—document level score}} \\ &\quad + \underbrace{\lambda_{T_{SQ}} \sum_{c \in T_{SQ}} f_{T_{SQ}}(c) + \lambda_{T_{SE}} \sum_{c \in T_{SE}} f_{T_{SE}}(c)}_{F_S(Q)+F_S(E)\text{—segment level score}} + \lambda_D f_D(D) \Bigg\}, \end{aligned} \tag{4.7}$$

where F_D and F_S are convenience functions defined by coarse-grained evidence oriented (at the document level) and fine-grained evidence oriented (at the segment level) components of the joint distribution, respectively. Furthermore, $F_D(Q)$ and $F_S(Q)$ are document and query dependent, while $F_D(E)$ and $F_S(E)$ are document and latent concept dependent. These will be used to simplify and clarify expressions derived throughout the remainder of this section.

Table 4.4 shows how $f_{T_{SQ}}$ and $f_{T_{SE}}$ are defined. The feature function $f_{T_{SQ}}$ estimates the relevance of a basic content unit (i.e., a segment) within the document to the query. It is based on three hypotheses: (1) if a document is relevant to the query, then a segment within the document is more likely to be relevant, (2) if a segment's neighboring segments are relevant to the query, then the segment itself is more likely to be relevant, and (3) the more query term occurrences a segment has, more likely the segment is likely to be relevant to the query. This is similar to the smoothing used by Murdock and Croft for sentence retrieval, which was shown

Table 4.4 Document structure feature functions used with LCE-HMRF. Here, $D^* = (S_{j-1}, S_j, D)$, and δ, ζ, $\tilde{\delta}$, and $\tilde{\zeta}$ are model hyperparameters that control smoothing for query term and latent concept features, respectively

Feature	Value
$f_{T_{SQ}}(q, D^*)$	$\log[(1-\delta-\zeta)\frac{tf_{q,S_j}}{\lvert S_j\rvert} + \delta\frac{tf_{q,S_{j-1}}}{\lvert S_{j-1}\rvert} + \zeta\frac{tf_{q,D}}{\lvert D\rvert}]$
$f_{T_{SE}}(e, D^*)$	$\log[(1-\tilde{\delta}-\tilde{\zeta})\frac{tf_{e,S_j}}{\lvert S_j\rvert} + \tilde{\delta}\frac{tf_{e,S_{j-1}}}{\lvert S_{j-1}\rvert} + \tilde{\zeta}\frac{tf_{e,D}}{\lvert D\rvert}]$

to be highly effective (Murdock and Croft 2005). Additionally, the feature function $f_{T_{SE}}$ measures how relevant the latent concept is with respect to the segment in a similar manner.

5.3 Discussion

Both LCE-GE and LCE-HMRF attempt to relax the independence assumption between query terms and latent concepts. LCE-GE *explicitly* incorporates the dependencies by linking query term nodes and the latent concept node on the graph. LCE-HMRF instead *implicitly* models these same dependencies via the segment nodes. In this way, the model can identify, and reward, expansion concepts that co-occur with one or more query terms within one or more segments. Modeling such co-occurrences on a per-segment basis, rather than at the document level provides an implicit means for modeling dependencies between query terms and expansion concepts.

By implicitly modeling the dependencies, the latent concept in LCE-HMRF now depends on the hierarchical structure within documents rather than the user query as in LCE-GE. As a result, LCE-HMRF does not have to evaluate expensive features defined over pairs of query terms and latent concepts like LCE-GE. In this way, LCE-HMRF can address LCE-GE's high computational complexity and data sparseness problems.

To illustrate how LCE-HMRF weights expansion concepts, we show the general form of the LCE conditional probabilities, which take on the following form:

$$P(E|Q) \approx \frac{\sum_{S_{j-1},S_j,D} P(Q, E, S_{j-1}, S_j, D)}{\sum_E \sum_{S_{j-1},S_j,D} P(Q, E, S_{j-1}, S_j, D)}$$
$$\propto \sum_{S_{j-1},S_j,D} \exp\{F_D(Q) + F_D(E) + F_S(Q) + F_S(E)\}, \qquad (4.8)$$

where $\{S_{j-1}, S_j, D\}$ is the set of triplets extracted from the set of relevant or pseudo-relevant documents R_Q for query Q. As we see, the contribution for each triplet is a combination of scores at the document level and scores at the segment level. The document level score reflects how relevant the original query is to the document,

as well as how relevant the expansion concept is. The same is true for the segment level score. As a consequence, an expansion concept that dominates segments of the document that are highly relevant to the original query will be assigned a high likelihood.

Based on the above analysis, the problem of assigning a score to a latent concept can be casted as two separate sub-problems, i.e. assigning relevance scores to basic content units (i.e. segments) within a document, and measuring how well a latent concept account for the basic content units. Since segments from different parts of a document can have very diverse semantics, the first sub-problem is non-trivial.

The bag of words models, together with LCE, explore co-occurrence at the document level only. Therefore, all segments within a document are assigned the same relevance score to the query. In this way, each term occurrence within the document *equally* contributes to the final score of the latent concept. Therefore, bad latent concepts that frequently occur in the irrelevant part of the document would be assigned high scores. Unlike these models, HMRFs can exploit a document's fine-grained evidence. Each segment is assigned different relevance score, totally based on its relevance to the query. Therefore, a term occurrence within the relevant segment contributes more to the final score of the latent concept.

We note that if we drop the dependence between adjacent segments from LCE-HMRF (i.e., we only consider a single segment S_j as evidence at a time), the model degrades into *segment-based LCE*, which we refer to as LCE-SB. With LCE-SB, documents are split into segments first, and then LCE is applied to segments for query expansion, treating each segment as a "document". This is akin to the passage-based query expansion performed by LCA (Xu and Croft 2000). While LCE-SB can exploit fine-grained document information, it cannot utilize valuable contextual information (e.g., information from sibling nodes). Thus, LCE-HMRF provides a generalization of passage-based query expansion approaches.

Moreover, when exploring co-occurrence at the segment level, there exists insufficient co-occurrence domain to accurately estimate a segment's topic distribution (e.g. the relevance score of the segment). Unlike LCE-SB, HMRFs consider triple-wise interactions, i.e. dependencies between neighboring sibling nodes and long distance dependencies between parent and child nodes. The triple-wise interactions provide a mechanism to utilize a document's structure information to smooth the segments.

5.4 Time Complexity

In this section, we compare the time complexity of LCE-GE and LCE-HMRF. The efficiency of LCE-GE's joint distribution computation in Eq. 4.6 is dominated by the clique sets dependent on the query, since clique sets that do not depend on the query can be precomputed at index-time. Among all the seven clique sets used for LCE-GE, the clique set U_{QE} is the most time-consuming one. For a query Q with $|Q|$ query terms, U_{QE} has $|Q| - 1$ cliques (b, E, D) within it, where b is a query

bigram. Given a fixed Q, the random variable E has $|V|$ (i.e., the size of the vocabulary) possible outcomes in total, and the random variable D has $|\mathcal{R}_Q|$ (i.e., the number of feedback documents) possible outcomes. Thus, the feature function $f_{U_{QE}}(b, E, D)$ needs to be computed $(|Q| - 1)\times|V|\times|\mathcal{R}_Q|$ times. Supposing the index contains term position information and a term occurs tf_{avg} times within a document on average, then $f_{U_{QE}}(b, E, D)$ must be computed $O(tf^3_{\text{avg}})$ times. As a result, the total complexity is $O(|Q|\cdot|V|\cdot|\mathcal{R}_Q|\cdot tf^3_{\text{avg}})$, where $|V|$, the vocabulary size, is often very large. Thus, LCE-GE suffers from high computational complexity problem.

Similarly, the efficiency of LCE-HMRF's joint distribution computation with Eq. 4.7 is dominated by the computation of $f_{U_D}(b, D)$ and $f_{T_{SQ}}(q, S_{j-1}, S_j, D)$. We see that neither of these feature functions depends on the latent concept node E, and thus they do not need to be computed once for every term in the vocabulary. Hence, the computational complexity of the model is significantly decreased compared to LCE-GE. The time complexity of computing $f_{U_D}(b, D)$ is $O(|Q|\cdot|\mathcal{R}_Q|\cdot tf^2_{\text{avg}})$, while the time complexity of $f_{T_{SQ}}(q, S_{j-1}, S_j, D)$ is $O(|Q|\cdot|\mathcal{R}_Q|\cdot tf_{\text{avg}}\cdot|S|_{\text{avg}})$ where $|S|_{\text{avg}}$ is the average number of segments per document.

5.5 HMRFs for Ranking

The HMRF model provides a natural, formally motivated mechanism for modeling term dependencies and document structure. Thus, we are interested in exploring how effective the model is for ranking compared to the original MRF model (Metzler and Croft 2005), independent of the query expansion task. As illustrated in Fig. 4.3, by removing the latent concept node from the model structure, the HMRF model defines a joint distribution over a query and a triplet of nodes in the document tree. Hence, the conditional probability of a triplet given the query can easily be computed and utilized for ranking. When HMRFs are used in this way (i.e., directly for ranking document instead of expanding queries), we refer to the resulting approach as Ranking-HMRF. In this work, we simply use the best match strategy to score a document for ranking, as follows:

$$\begin{aligned} Score(Q, D) &= \max\{P(S_{j-1}, S_j, D|Q)\} \\ &\overset{\text{rank}}{=} \max\{P(Q, S_{j-1}, S_j, D)\}, \end{aligned} \tag{4.9}$$

where $\{S_{j-1}, S_j, D\}$ is the set of triplets extracted from document D, and $P(Q, S_{j-1}, S_j, D)$ is the joint distribution of Ranking-HMRF.

We compare the effectiveness of Ranking-HMRF with the original MRF model in Table 4.5. Both MRF and Ranking-HMRF employ the sequential dependency assumption, which assumes that dependencies exist between adjacent query terms. For efficiency reasons, we conduct the Ranking-HMRF approach by re-ranking the

Table 4.5 Mean average precision for LM, MRF, and Ranking-HMRF. The superscripts α and β indicate statistically significant improvements ($p < 0.05$ using Wilcoxon test) over LM and MRF, respectively

	LM	MRF	Ranking-HMRF
ROBUST04	0.2532	0.2653^{α}	$0.2684^{\alpha\beta}$
WT10g	0.1968	0.2074^{α}	$0.2144^{\alpha\beta}$
GOV2	0.2981	0.3244^{α}	$0.3332^{\alpha\beta}$

top 3000 results returned by the baseline approach (i.e., unigram language model (Ponte and Croft 1998)).

As shown in the table, Ranking-HMRF consistently and significantly outperforms the MRF model across all the data sets. We attribute the additional performance gain to the hierarchical document structure which is naturally captured by Ranking-HMRF. Basically, the original MRF model can capture two kinds of features, namely term occurrence and term proximity features. The former features are loosely defined at the document level while the latter ones are strictly defined within a limited window size (e.g., 8 terms). The document structure aware Ranking-HMRF approach provides a way for exploring co-occurrence at different spatial scales, and can define features at segment level which are less strict than the proximity features and more focused than the term occurrence features.

6 Experimental Results

In order to better understand the strengths and weaknesses of the technique, we evaluate it on a wide range of data sets. Appendix A provides a summary of the TREC data sets considered. For each data set, we split the available topics into a training and test set, where the training set is used solely for parameter estimation and the test set is used for evaluation purposes.

In all cases, only the title portion of the TREC topics are used to construct queries. We construct G using the sequential dependence assumption for all data sets (Metzler and Croft 2005).

6.1 Ad Hoc Retrieval Results

We now investigate how well the model performs in practice in a pseudo-relevance feedback setting. We compare the MRF-FI model, the MRF-SD model (without expansion), relevance models, and LCE to better understand how each model performs across the various data sets.

For the MRF models, we train the model parameters (i.e., Λ) and model hyperparameters (i.e., μ^t, μ^w). For RM3 and LCE, we also train the number of pseudo-relevant feedback documents used and the number of expansion terms.

Table 4.6 Test set mean average precision for MRF-FI, MRF-SD, relevance models (RM3), and latent concept expansion (LCE). The superscripts α, β, and γ indicate statistically significant improvements over MRF-FI, MRF-SD, and RM3, respectively

	MRF-FI	MRF-SD	RM3	LCE
AP	0.2077	0.2147^{α}	$0.2518^{\alpha\beta}$	$0.2692^{\alpha\beta\gamma}$
WSJ	0.3258	0.3425^{α}	0.3493^{α}	$0.3943^{\alpha\beta\gamma}$
ROBUST04	0.2920	0.3096^{α}	$0.3382^{\alpha\beta}$	$0.3601^{\alpha\beta\gamma}$
WT10g	0.1861	0.2053^{α}	0.1944^{α}	$0.2269^{\alpha\beta\gamma}$
GOV2	0.3234	0.3520^{α}	0.3656^{α}	$0.3924^{\alpha\beta\gamma}$

6.1.1 Expansion with Single Term Concepts

We begin by evaluating how well the model performs when expanding using only single terms. The expansion term likelihoods are computed according to Eq. 4.4. The equation clearly shows how LCE differs from relevance models. As we stated before, when we set $\lambda_{T_D} = \lambda'_{T_D} = 1$ and all other parameters to 0, we obtain the exact formula that is used to compute term likelihoods in the relevance modeling framework. Therefore, LCE adds two very important factors to the equation. First, it adds the ordered and unordered window features that are applied to the original query. Second, it applies an intuitive *tf.idf*-like form to the candidate expansion term w. The *idf* factor, which is not present in relevance models, plays an important role in expansion term selection.

The results, evaluated using mean average precision, are given in Table 4.6. As we see, the MRF-SD model, relevance models, and LCE always significantly outperform the bag of words MRF-FI model. In addition, LCE shows significant improvements over relevance models across all data sets. The relative improvements over relevance models is 6.9% for AP, 12.9% for WSJ, 6.5% for ROBUST04, 16.7% for WT10G, and 7.3% for GOV2.

We also note the interesting result that the MRF-SD model is statistically equivalent to relevance models on the two Web data sets. In fact, the MRF-SD model outperforms relevance models on the WT10g data set. This reiterates the importance of non-unigram, proximity-based features for content-based Web search observed previously (Metzler and Croft 2005; Metzler et al. 2005b).

We also evaluate the methods according to precision at 10 in Table 4.7. Not surprisingly, the precision at 10 improvements achieved using LCE were not as significant as those observed when using mean average precision. It is well understood that expansion is recall enhancing, rather than precision enhancing. The MRF model significantly improves upon the unigram language model on every data set, but only in a few cases do relevance models or LCE significantly improve upon the MRF model.

It should also be noted that, despite the fact that the model has more free parameters than relevance models, there is surprisingly little overfitting. Instead, it exhibits good generalization properties.

Table 4.7 Test set precision at 10 for MRF-FI, MRF-SD, relevance models (RM3), and latent concept expansion (LCE). The superscripts α, β, and γ indicate statistically significant improvements over MRF-FI, MRF-SD, and RM3, respectively

	MRF-FI	MRF-SD	RM3	LCE
AP	0.3460	0.3340	0.3640^{β}	0.3720^{β}
WSJ	0.4860	0.5080	0.4980	$0.5400^{\alpha\beta\gamma}$
ROBUST04	0.4293	0.4566^{α}	0.4576^{α}	$0.4798^{\alpha\gamma}$
WT10g	0.3204	0.3245	0.3265	$0.3633^{\alpha\beta\gamma}$
GOV2	0.5180	0.5680^{α}	0.6160^{α}	0.6060^{α}

Table 4.8 Test set mean average precision values for multi-term concept LCE experiments

	One Term	One + Two Term
AP	0.2692	0.2648
WSJ	0.3943	0.3945
ROBUST04	0.3601	0.3464
WT10G	0.2269	0.2187
GOV2	0.3924	0.3900

Table 4.9 One and two term expansion concepts for the query *price fixing* (ROBUST04 topic 622) and *tax evasion indicted* (ROBUST04 topic 650). Concepts are listed in descending order of $P(e|Q)$ and $P(e_1, e_2|Q)$, respectively

Price fixing		Tax evasion indicted	
price	fixed pricing	evasion	tax evasion
fixed	price fixing	tax	income tax
market	control	indicted	tax charges
report	market price	federal	los angeles
vote	united states	charges	san diego

6.1.2 Expansion with Multi-term Concepts

We also investigated expanding using both single and two word concepts. For each query, we expanded using a set of single term concepts, as well as a set of two term concepts. The sets were chosen independently. The results of the experiments are shown in Table 4.8. As the results show, there is little or no improvement when including two term concepts. In fact, the results are statistically indistinguishable for all data sets.

This unexpected result may be due to the fact that strong correlations exist between the single term expansion concepts. We found that the two word concepts chosen often consisted of two highly correlated terms that are also chosen as single term concepts. For example, for the query *price fixing*, there was a great deal of

redundancy. Table 4.9 shows the single and two term concepts that were selected for expansion. We see "fixed pricing" and "price fixing", "price", and "fixed" all occur in the list. While these are excellent expansion terms, the fact that they appear so many times likely results in terms "price" and "fixing" being overemphasized. Additionally, this query adds "control control" and "united states", which are poor expansion concepts. These two concepts are given high probability because of the features used within the model. A similar analysis holds for the query *tax evasion indicted*. Therefore, other feature sets may ultimately yield different results, especially if they reduce the correlation among the expansion concepts. While the IDF-like feature may be appropriate for choosing single terms, it may be better to apply some other type of feature to the ordered and unordered cliques within the graph. A better understanding of phrases and named entities (e.g., San Diego, Los Angeles, United States in Table 4.9), and their effect on query expansion is still needed.

Furthermore, due to the computational complexity involved in computing multi-term expansion, we had to limit the number of multi-term expansion concepts that we used. We were unable to perform a full sweep over all such values. However, we made sure to focus on the most promising values, and, based on previous experience, it is unlikely that the results would have been significantly better than the one term expansion concepts, but perhaps would have been slightly improved over their current value.

Therefore, the experiments yield no conclusive results with regard to expansion using multi-term concepts. Instead, the results introduce interesting open questions and directions for future exploration.

6.2 Robustness

As we have shown, relevance models and latent concept expansion can significantly improve retrieval effectiveness over the baseline MRF-FI model. In this section we analyze the robustness of these two methods.

Figure 4.4 provides an analysis of the robustness of relevance modeling and latent concept expansion for the AP, WSJ, ROBUST04, and WT10G data sets. The analysis for GOV2 is similar. The histograms provide, for various ranges of relative decreases/increases in mean average precision, the number of queries that were hurt/improved with respect to the MRF-FI baseline.

As the results show, LCE exhibits strong robustness for each data set. For AP, relevance models improve 38 queries and hurt 11, whereas LCE improves 35 and hurts 14. Although relevance models improve the effectiveness of 3 more queries than LCE, the relative improvement exhibited by LCE is significantly larger. For the ROBUST04 data set, relevance models improve 67 queries and hurt 32, and LCE improves 77 and hurts 22. Finally, for the WT10G collection, relevance models improve 32 queries and hurt 16, and LCE improves 35 and hurts 14. As with AP, the amount of improvement exhibited by the LCE versus relevance models is significantly larger for both the ROBUST04 and WT10G data sets. In addition, when

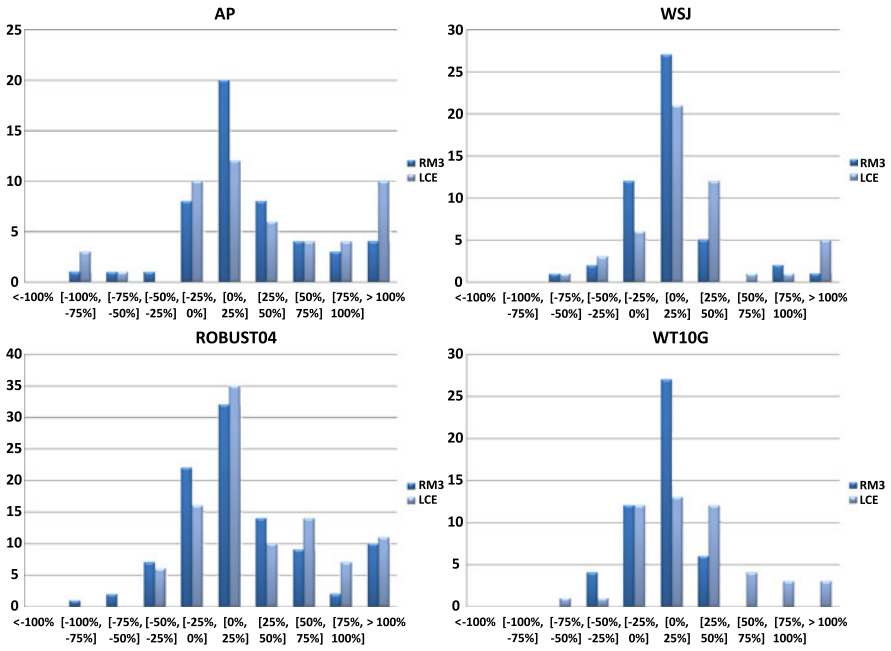

Fig. 4.4 Histograms that demonstrate and compare the robustness of relevance models (RM3) and latent concept expansion (LCE) with respect to the MRF-FI model for the AP, WSJ, ROBUST04, and WT10G data sets

LCE does hurt performance, it is less likely to hurt as much as relevance modeling, which is a desirable property.

Overall, LCE improves effectiveness for 65%–80% of queries, depending on the data set. When used in combination with a highly accurate query performance prediction system, it may be possible to selectively expand queries and minimize the loss associated with sub-baseline performance.

6.3 Multi-term Concept Generation

Although we found that expansion using multi-term concepts failed to produce conclusive improvements in effectiveness, there are other potential tasks that these concepts may be useful for, such as query suggestion/reformulation, summarization, and concept mining. For example, for a query suggestion task, the original query can be used to generate a set of latent concepts which correspond to alternative query formulations.

Although evaluating the model on these tasks is beyond the scope of this work, we wish to show an illustrative example of the types of concepts generated using the model. In Table 4.10, we present the most likely one, two, and three term concepts

Table 4.10 Fifteen most likely one, two, and three word concepts constructed using the top 25 documents retrieved for the query *hubble telescope achievements* on the ROBUST04 collection

1 word concepts	2 word concepts	3 word concepts
telescope	hubble telescope	hubble space telescope
hubble	space telescope	hubble telescope space
space	hubble space	space telescope hubble
mirror	telescope mirror	space telescope NASA
NASA	telescope hubble	hubble telescope astronomy
launch	mirror telescope	NASA hubble space
astronomy	telescope NASA	space telescope mirror
shuttle	telescope space	telescope space NASA
test	hubble mirror	hubble telescope mission
new	NASA hubble	mirror
discovery	telescope astronomy	space telescope launch
time	telescope optical	space telescope discovery
universe	hubble optical	shuttle space telescope
optical	telescope discovery	hubble telescope flaw
light	telescope shuttle	two hubble space

generated using LCE for the query *hubble telescope achievements* using the top 25 ranked documents from the ROBUST04 collection.

It is well known that generating multi-term concepts using a unigram-based model produces unsatisfactory results, since it fails to consider term dependencies. This is not the case when generating multi-term concepts using the model. Instead, a majority of the concepts generated are well-formed and meaningful. There are several cases where the concepts are less coherent, such as *mirror*. In this case, the likelihood of the term *mirror* appearing in a pseudo-relevant document outweighs the ICF features, which causes this non-coherent concept to have a high likelihood. Such examples are in the minority, however.

Not only are the concepts generated well-formed and meaningful, but they are also topically relevant to the original query. As we see, all of the concepts generated are on topic and in some way related to the Hubble telescope. It is interesting to see that the concept *hubble telescope flaw* is one of the most likely three term concepts, given that it is somewhat contradictory to the original query. Despite this contradiction, documents that discuss the telescope flaws are also likely to describe the successes, as well, and therefore this is likely to be a meaningful concept.

One important thing to note is that the concepts LCE generates are of a different nature than those that would be generated using a bigram relevance model. For example, a bigram model would be unlikely to generate the concept *telescope space NASA*, since none of the bigrams that make up the concept have high likelihood. However, since the model is based on a number of different features over various types of cliques, it is more general and robust than a bigram model.

Although we only provided the concepts generated for a single query, we note that the same analysis and conclusions generalize across other data sets, with coherent, topically related concepts being consistently generated using LCE.

6.4 Evaluation of LCE-GE and LCE-HMRF

Earlier in this chapter we described two LCE extensions called generalized LCE (LCE-GE) and LCE using HRMFs (LCE-HMRF). Unlike basic LCE, both of these approaches modeled the dependencies between query terms and expansion concepts. In this section, we evaluate the effectiveness of these approaches empirically. Experimental results show that these term dependencies can help improve the retrieval performance consistently and significantly during expansion.

6.4.1 Experimental Setup

Six different models are compared in the study, including: the unigram language modeling with Dirichlet smoothing (LM), the relevance model (RM3); latent concept expansion (LCE), the generalized LCE model (LCE-GE), LCE using Hierarchical Markov random fields (LCE-HMRF), and segment-based LCE (LCE-SB).

We compare LM, RM3, LCE, and LCE-HMRF to better understand how modeling term dependencies can contribute to retrieval performance during expansion. Additionally, we compare LCE-GE and LCE-HMRF to evaluate whether models that explicitly model dependencies between query terms and expansion terms are more effective than those that implicitly model such dependencies. Finally, we compare LCE-SB and LCE-HMRF to evaluate the usefulness of the segment-level interactions considered by LCE-HMRF compared to a simpler passage-based expansion approach.

Unlike the previous experiments, which used a fixed training and test split, we utilize cross-validation to estimate the parameters and evaluate the results for each data set. Given a data set, topics are divided into subsets of 50 topics each. Each subset, in turn, is used as a testing set, while the rest of the topics serve as a training set. Experiments are run separately for each data set, and average results over all testing sets are reported.

For the unigram language model, the smoothing parameter is trained. For RM3, LCE, LCE-GE, we train the model parameters, model hyperparameters, the number of pseudo-relevant documents used, and the number of expansion terms. For LCE-HMRF and LCE-SB, we also train the segment length. Meanwhile, LCE-HMRF adopts the same first-pass and second-pass retrieval algorithm (i.e. the MRF model for information retrieval (Metzler and Croft 2005)) as LCE, where the only difference between LCE-HMRF and LCE is how each selects and weights expansion terms.

Table 4.11 Mean average precision for language modeling (LM), relevance model (RM3), latent concept expansion (LCE), and LCE using Hierarchical Markov random fields (LCE-HMRF). The superscripts α, β, and γ indicate statistically significant improvements ($p < 0.05$ using Wilcoxon test) over LM, RM3, LCE, respectively

	LM	RM3	LCE	LCE-HMRF
ROBUST04	0.2532	0.2834^{α}	$0.3057^{\alpha\beta}$	$0.3313^{\alpha\beta\gamma}$
WT10g	0.1968	0.2118^{α}	$0.2259^{\alpha\beta}$	$0.2454^{\alpha\beta\gamma}$
GOV2	0.2981	0.3179^{α}	$0.3454^{\alpha\beta}$	$0.3634^{\alpha\beta\gamma}$

6.4.2 Basic Results

In this section, we empirically evaluate the following hypothesis: Query expansion approaches that properly consider term dependencies will perform better than approaches that do not consider all of dependencies essential for the query expansion task (i.e., the dependencies between query terms and the dependencies between query terms and expansion terms).

The mean average precision for LM, RM3, LCE, and LCE-HMRF are given in Table 4.11. As would be expected, the relevance model, LCE, and LCE-HMRF always significantly outperform the unigram language model.

Furthermore, LCE significantly outperforms the relevance model across all data sets. This indicates that term dependencies can provide extra information to help estimate a better query model than using independent terms alone and can contribute to the retrieval performance during expansion. The findings agree with similar experiments that were previously carried out (Metzler and Croft 2007).

As the results show, by implicitly incorporating term dependencies between query terms and expansion terms, LCE-HMRF achieves the best performance across all of the data sets. In particular, it achieves 16.9%, 15.9%, and 14.3% relative improvements in mean average precision over the relevance model, on ROBUST04, WT10g, and GOV2, respectively. Moreover, LCE-HMRF shows significant improvements over the original LCE approach, which demonstrates the benefits of using the document structure to implicitly model dependencies between query terms and expansion terms during expansion and support the hypothesis. The relative improvements over LCE are 8.4% for ROBUST04, 8.6% for WT10g, and 5.2% for GOV2. All of the improvements, with respect to relevance models and LCE are statistically significant.

6.4.3 Robustness

As we have observed, the relevance model, LCE, and LCE-HMRF can lead to significantly improvements in retrieval effectiveness on average versus a simple unigram language modeling baseline. Here, we demonstrate and compare the robustness of these query expansion techniques with respect to this baseline. We define robustness as the number of queries whose effectiveness is improved/degraded (and by

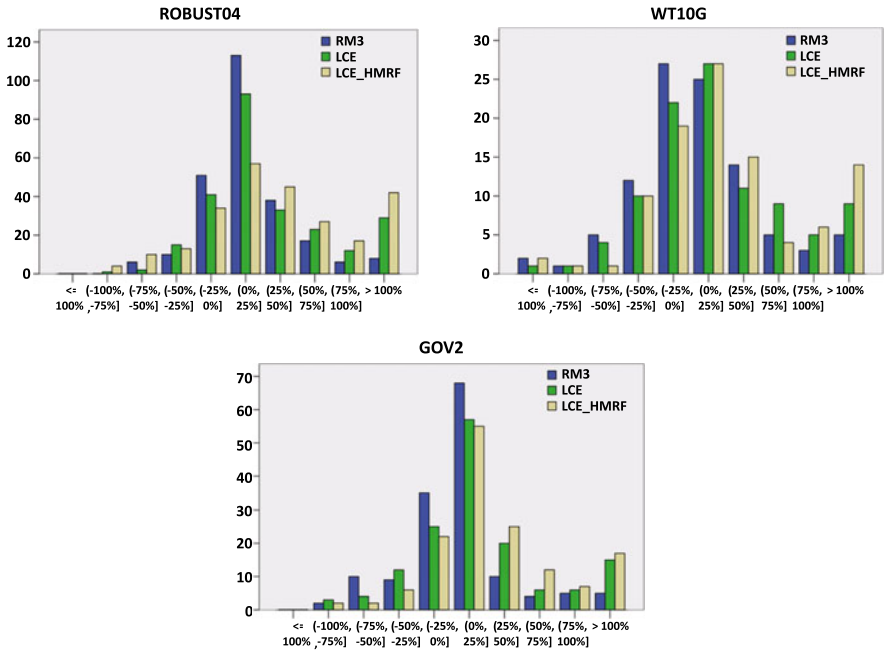

Fig. 4.5 Histograms that compare the robustness of the relevance model (RM3), latent concept expansion (LCE), and LCE using Hierarchical Markov random fields (LCE-HMRF) with respect to the unigram language model (LM) for the ROBUST04, WT10g, and GOV2 data sets

how much) as the results of applying these methods. A highly robust expansion technique will significantly improve many queries and only minimally degrade very few.

Figure 4.5 presents an analysis of the robustness of the relevance model, LCE, and LCE-HMRF on ROBUST04, WT10g, and GOV2. The histograms present, for various ranges of relative decreases/increases in mean average precision, the number of queries that are hurt/improved over the baseline unigram language model.

The models that consider term dependencies show strong robustness for each data set. For the ROBUST04 data set, the relevance model improves 182 queries and degrades 67, whereas LCE improves 190 and degrades 59 and LCE-HMRF improves 188 and degrades 61. Although LCE improves the effectiveness of 2 more queries than LCE-HMRF, the relative improvement exhibited by LCE-HMRF is significantly larger. For WT10g, the relevance model improves 52 and degrades 47, while LCE improves 61 and degrades 38 and LCE-HMRF improves 66 and degrades 33. Finally, for GOV2, the relevance model improves 92 and degrades 56, while LCE improves 104 and degrades 44 and LCE-HMRF improves 116 and degrades 32. Therefore LCE-HMRF exhibits good robustness characteristics.

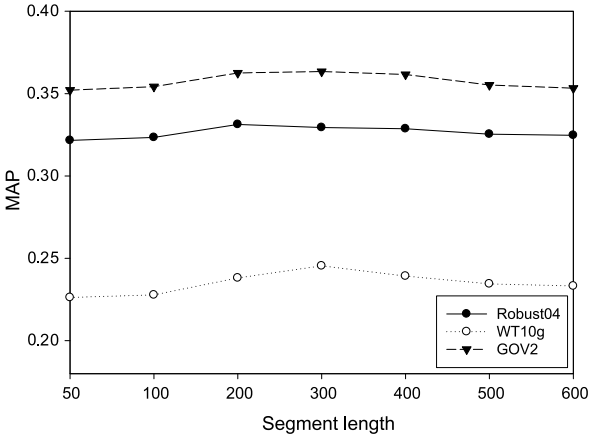

Fig. 4.6 Sensitivity to segment length

6.4.4 Parameter Sensitivity

Finally, we note that the HMRF model relies on automatically breaking documents into segments. Therefore, we are interested in analyzing the sensitivity of the retrieval performance with respect to the size of the segment. This sensitivity analysis is shown in Fig. 4.6. For each segment length, we train all of the other model parameters to fairly evaluate the sensitivity. The results show that the effectiveness is relatively stable across different segment lengths. However, for ROBUST04, the homogeneous newswire data set, *segment length* = 200 performed the best. For the heterogeneous Web data sets such as WT10g and GOV2, *segment length* = 300 performed the best. These results suggest that setting the segment length in the range of 200–300 is a reasonable 'default' setting, at least for the data sets currently under consideration.

6.4.5 LCE-GE vs. LCE-HMRF

In this section, we empirically evaluate the following hypothesis: Hierarchical models that seamlessly integrate the hierarchical document structure can address the high computational complexity and data sparseness problems for modeling term dependencies, which are suffered by non-hierarchical models.

We compare the effectiveness of LCE-GE, which is an non-hierarchical model and explicitly models the dependencies between query terms and expansion concepts, and LCE-HMRF, which models the same type of dependencies implicitly via the hierarchical document structure.

The results of this comparison are provided in Table 4.12. The results show that LCE-HMRF significantly outperforms LCE-GE on all data sets. The GOV2 data set is not tested due to the high computational cost involved in applying LCE-GE. Interestingly, LCE-GE shows improvements over LCE, but the improvements are not statistically significant. Although LCE-GE and LCE-HMRF both model dependencies between query terms and expansion concepts, LCE-GE is shown to not be

Table 4.12 Mean average precision for LCE, LCE-GE, LCE-HMRF. The superscripts α and β indicate statistically significant improvements ($p < 0.05$ using Wilcoxon test) over LCE and LCE-GE, respectively

	LCE	LCE-GE	LCE-HMRF
ROBUST04	0.3057	0.3092	$0.3313^{\alpha\beta}$
WT10g	0.2259	0.2287	$0.2454^{\alpha\beta}$

Table 4.13 Mean average precision for LCE, LCE-SB, and LCE-HMRF. The superscripts α and β indicate statistically significant improvements ($p < 0.05$ using Wilcoxon test) over LCE and LCE-SB, respectively

	LCE	LCE-SB	LCE-HMRF
ROBUST04	0.3057	0.2981	$0.3313^{\alpha\beta}$
WT10g	0.2259	0.2182	$0.2454^{\alpha\beta}$
GOV2	0.3454	0.3511	$0.3634^{\alpha\beta}$

as effective as LCE-HMRF, presumably due to the data sparseness issue described earlier.

Additionally, the computational cost of LCE-GE is much higher than LCE-HMRF. Unlike LCE-HMRF, LCE-GE explicitly models a large number of pairs of query terms and latent concepts. Thus, LCE-GE has to compute many term proximity features over these pairs, which is very time consuming. Therefore, LCE-HMRF is a practically appealing query expansion approach, both in terms of efficiency and effectiveness. Of course, in a real, large-scale system, query expansion would likely be either done offline or done online using a hybrid strategy, such as the one described by Broder et al. (2009). However, even when such an architecture is used, LCE-HMRF is still desirable to LCE-GE. As a consequence, we can support the hypothesis.

6.4.6 Segment-Based LCE vs. LCE-HMRF

As mentioned earlier, the dependencies between sibling nodes and the parent node considered by the HMRF model provides a mechanism for utilizing valuable contextual information inside a document. To quantify the utility of these dependencies, we compare the effectiveness of LCE-SB (passage-based LCE) and LCE-HMRF.

As shown in Table 4.13, LCE-HMRF always significantly outperforms LCE-SB. In fact, LCE-SB shows slightly lower performance than LCE on the ROBUST04 and WT10g data sets. Although LCE-SB has the capability to exploit a document's fine-grained evidence, it fails to significantly improve the retrieval performance over LCE, likely due to data sparseness, which stems from the fact that LCE-SB only considers information from a single segment, instead of from two adjacent segments and the entire document, as is the case in LCE-HMRF. The segment-level dependencies considered by LCE-HMRF can help alleviate this problem by leveraging the

document's structure information to smooth the segments and improve the retrieval performance. Therefore, LCE-HMRF is more than simply a passage-based version of LCE, as these results clearly show that modeling intra-segment dependencies is highly effective.

7 Discussion

We conclude this chapter by discussing several theoretical issues involved with regard to query expansion.

7.1 Relevance vs. Relevant Documents

There are other reasonable ways of using the MRF model for query expansion beyond using Eq. 4.3. For example, given a set of relevant or pseudo-relevant documents $D_1, \ldots, D_n$, we could generate expansion concepts by computing $P(E|D_1, \ldots, D_n)$.

The concepts generated in this way tend to show excellent effectiveness on the training set. However, they fail to yield significant improvements in retrieval effectiveness on the test set. In fact, using this expansion model shows little-to-no improvements over relevance models. It is somewhat surprising that this formulation performs so much worse on the test set. Buckley and Salton noted a similar observation when optimizing term weights using a Rocchio-based expansion technique (Buckley and Salton 1995). They argued that there is a subtle, yet important difference between modeling relevance and modeling relevant documents.

Under this alternative formulation, we are directly modeling relevant documents. By doing so, we are overfitting the model. This explains the poor generalization behavior exhibited using this alternative formulation. This, however, is not an issue when using the original formulation. Using that formulation, $P_{H,\Lambda}(E|Q)$ can be thought of as a model of relevance, since it generates terms that are highly likely given the query, which is the best evidence of relevance. This avoids the problem of overfitting the model to the relevant documents by modeling a more generalized notion of relevance.

Therefore, the results obtained by Salton and Buckley using the Rocchio-based expansion technique apply to statistical query expansion techniques, as well. Thus, future query expansion models, to avoid overfitting, should focus on modeling more generalized notions of relevance, rather than the relevant documents themselves.

7.2 The Role of Dependence

Our latent concept expansion technique captures two semi-orthogonal types of dependence. In information retrieval, there has been a long-term interest in under-

standing the role of term dependence. Out of this research, two broad types of dependencies have been identified.

The first type of dependence is *syntactic dependence*. This type of dependence covers phrases, term proximity, and term co-occurrence (Clarke and Cormack 2000; Croft 1986; Croft et al. 1991; Fagan 1987; van Rijsbergen 1977). These methods capture the fact that queries implicitly or explicitly impose a certain set of positional dependencies.

The second type is *semantic dependence*. Examples of semantic dependence are relevance feedback, pseudo-relevance feedback, synonyms, and to some extent stemming (Collins-Thompson and Callan 2005). These techniques have been explored on both the query and document side. On the query side, this is typically done using some form of query expansion, such as relevance models or LCE. On the document side, this is done as document expansion or document smoothing (Kurland and Lee 2004; Liu and Croft 2004; Tao et al. 2006).

Although there may be some overlap between syntactic and semantic dependencies, they are mostly orthogonal. The model uses both types of dependencies. The use of phrase and proximity features within the model captures syntactic dependencies, whereas LCE captures query-side semantic dependence. This explains why the initial improvement in effectiveness achieved by using the MRF model is not lost after query expansion. If the same types of dependencies were captured by both syntactic and semantic dependencies, LCE would be expected to perform about equally as well as relevance models. Therefore, by modeling both types of dependencies we see an additive effect, rather than an absorbing effect.

An interesting area of future work is to determine whether or not modeling document-side semantic dependencies can add anything to the model. Previous results that have combined query- and document-side semantic dependencies have shown mixed results (Liu and Croft 2004; Wei and Croft 2006).

Chapter 5
Query-Dependent Feature Weighting

1 Overview

Most traditional information retrieval models, such as language modeling and BM25, utilize very simple user query models. These models tend to treat query terms as independent and of uniform importance. Simple heuristics, such as inverse document frequency (*idf*), are integral parts of these models and can be thought of as a simple query term weighting model, but they are very rigid and are based on a single data source. Furthermore, it is not clear if *idf* is an appropriate measure of importance for phrases and other generic concepts (Pickens and Croft 1999). Recent research has shown that modeling query term dependencies and using non-uniform query term weighting (beyond *idf*) can significantly improve retrieval effectiveness, especially on very large collections and for long, complex queries (Bendersky and Croft 2008; Lease 2009; Metzler and Croft 2005).

This chapter extends the basic MRF model by automatically learning query-dependent concept weights. The extension is a generic framework for learning the importance of query term concepts in a way that directly optimizes an underlying retrieval metric. It is important to note that this is quite different from query segmentation approaches (Bergsma and Wang 2007; Guo et al. 2008; Tan and Peng 2008). Optimizing segmentation accuracy is not guaranteed to optimize retrieval effectiveness. By implementing concept weighting directly into the underlying retrieval model we avoid the issue of metric divergence (Morgan et al. 2004). As we will show, this strategy yields strong retrieval effectiveness gains.

As an illustration of such metric divergence, Table 5.1 shows an actual example of unigram and bigram concept importances learned within the model for the query "civil war battle reenactments". If, instead, the weighting was done based on the output of a query segmenter, then it is likely that the phrase "civil war" and perhaps "battle reenactments" would be given large weights. However, the model assigns high weights to the unigram "reenactments" and the bigram "war battle", which happen to be the most *discriminative* (between relevant and non-relevant documents) concepts, not the *most likely* concepts in terms of query segmentation.

The remainder of this chapter, which is based on Bendersky et al. (2010), describes one possible approach for incorporating query-dependent weighting into the

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8_5,

Table 5.1 Query-dependent concept weights generated for query "civil war battle reenactments"

Concept	Weight
civil	0.0619
war	0.1947
battle	0.0913
reenactments	0.3487
civil war	0.1959
war battle	0.2458
battle reenactments	0.0540

basic MRF framework. The chapter begins by laying out the theoretical foundations and concludes with a detailed empirical evaluation that demonstrates the practical utility of the approach.

2 Related Work

Modeling atomic query concepts through term dependencies, or proximities, proved to have a significant positive impact on retrieval effectiveness on both TREC and Web corpora (Bai et al. 2008; Bendersky et al. 2009; Cummins and O'Riordan 2009; Metzler and Croft 2005; Mishne and de Rijke 2005; Tao 2007). Most of this work, however, was restricted to modeling term dependencies, rather than weighting them. In other words, all concept matches in the query had the same impact on the document score. While this assumption is reasonable for short keyword queries, it is much less reasonable for longer, more complex queries.

Recent work, focused on verbose queries, started to explore the direction of assigning varying *document independent* weights to query concepts. Kumaran and Carvalho (2009) address this by automatically removing extraneous terms that may have a negative effect on the overall retrieval performance of a query. Bendersky and Croft (2008) use a supervised discovery method for "key concepts" in verbose queries, and use a ranking approach that integrates the weighted key concepts with the original query. They find that weighted key concepts approach outperforms the standard bag-of-words model, however its performance is on par with the sequential dependence model that does not use any concept weighting (Metzler and Croft 2005). Most recently, Lease (2009) extended his previous work on term weighting (Lease et al. 2009) to show that incorporating learned term weights in a sequential dependence model improves the retrieval performance over the unweighted variant for verbose description queries on a number of TREC collections.

An additional information retrieval method that is related to the work described in this chapter is pseudo relevance feedback (PRF), which can be viewed as both query expansion and query term weighting technique (Lavrenko and Croft 2001). Recently, researchers separately focused on both modeling term dependencies (Metzler and Croft 2007) and term weighting (Cao et al. 2008) within the PRF framework.

There are two major differences between the model we describe here and the aforementioned previous work. First, the extension of the basic MRF model described here provides a principled retrieval framework that, unlike previously proposed methods, naturally combines both term and phrase weights. Second, the model parameters are estimated by directly maximizing the underlying retrieval metric. This differentiates the model described here from previous methods for concept weighting that employed indirect parameter estimation, maximizing metrics not directly related to retrieval performance such as classification accuracy (Bendersky and Croft 2008; Cao et al. 2008) or expected query model performance (Lease et al. 2009; Lease 2009). We show that the direct optimization approach allows us to achieve consistent performance gains over a range of query types, while previous work on concept weighting was mainly concentrated on verbose (Bendersky and Croft 2008; Lease 2009) or expanded (Cao et al. 2008) queries.

Direct optimization of an underlying retrieval metric ties the model described in this chapter to learning-to-rank approaches for information retrieval (LR4IR) (see Liu 2009 for a recent survey). The formulation of metric optimization is similar to some previous work, and thus allows us to build upon the existing direct optimization methods, such as those covered in Chap. 6. The primary benefit of the method lies in the fact that we are not limited to a linear combination of pairwise query-document features, as is usually the case in LR4IR (Liu 2009). Instead, we can also use *individual concept features* to effectively learn a concept weighting model in a similar, yet much more flexible, way than that proposed by Gey (1994). As we will show, this approach allows us to improve upon retrieval models that use only query-document dependent features.

3 Weighted Dependence Model

One of the primary limitations of the sequential dependence variant of the MRF model is the fact that all matches of the same type (e.g., term, ordered window, or unordered window) are treated as being equally important. This is the result of the massive parameter tying that is done with the sequential dependence model. Instead, it would be desirable to weight, *a priori*, different terms (or bigrams) within the query differently based on *query-level evidence*. For example, in a verbose query, there will likely be a few concepts (terms or phrases) within the query that will carry the most weight. While the sequential dependence model would treat all of the concepts as equally important, we would like to be able to weight the concepts appropriately, with regard to each other.

There are several ways to model this in the MRF model, but perhaps the most straightforward is to allow the parameters λ to depend on the concept that they are being applied to, rather than some global weight. This can be achieved by defining the potentials within the model as follows:

$$\psi(q_i, D; \Lambda) = \exp\left[\lambda(q_i) f_T(q_i, D)\right], \tag{5.1}$$

$$\psi(q_i, q_{i+1}, D; \Lambda) = \exp\big[\lambda(q_i, q_{i+1}) f_O(q_i, q_{i+1}, D) + \lambda(q_i, q_{i+1}) f_U(q_i, q_{i+1}, D)\big], \tag{5.2}$$

where $\lambda(q_i)$ is a parameter that depends on term q_i and $\lambda(q_i, q_{i+1})$ is a parameter that depends on the bigram q_i, q_{i+1}. In this setting, each term and bigram has a separate weight associated with it that is *independent of the document*. This parameter should be some measure of the general importance of the concept with respect to the rest of the query.

Although this formulation of the model is more general, it results in an infeasibly large number of parameters, since each λ now depends on the identity of one (or two) query terms. This was not a problem in the original formulation of the sequential dependence model, because it was assumed that all of the λ parameters, for a given match type, were tied to the same value, resulting in just three parameters. The solution described here is in the middle ground between these two extremes. We assume that the parameters λ take on a parameterized form. For simplicity, we assume the following weighted linear form:

$$\lambda(q_i) = \sum_{j=1}^{k_u} w_j^u g_j^u(q_i), \tag{5.3}$$

$$\lambda(q_i, q_{i+1}) = \sum_{j=1}^{k_b} w_j^b g_j^b(q_i, q_{i+1}), \tag{5.4}$$

where $g^u(q_i)$ and $g^b(q_i, q_{i+1})$ are features defined over unigrams and bigrams, respectively. Similarly, w^u and w^b are free parameters that must be estimated. If there are k_u unigram features and k_b bigram features, then we have a total of $k_u + k_b$ total parameters to estimate, compared to three in the sequential dependence model. The features $g^u(q_i)$ and $g^b(q_i, q_{i+1})$ are document independent and should be useful for determining the relative importance of the concept within the context of the query.

When the parameters λ have this parametric form, the final MRF ranking function can be shown to have the following form:

$$\begin{aligned} P(D|Q) \overset{\text{rank}}{=} & \sum_{i=1}^{k_u} w_i^t \sum_{q \in Q} g_i^u(q) f_T(q, D) \\ & + \sum_{i=1}^{k_b} w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_O(q_j, q_{j+1}, D) \\ & + \sum_{i=1}^{k_b} w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_U(q_j, q_{j+1}, D) \end{aligned} \tag{5.5}$$

which we call the *weighted sequential dependence model*. It is important to note that this model can easily be extended to handle other dependence assumptions,

Table 5.2 Statistics used to estimate term importance for a concept e. Concept e is either a query term q_i or a sequential query term pair $q_i q_{i+1}$

Data Source	Feature	Description
`Collection`	cf_e	Collection frequency for concept e
	df_e	Document frequency for concept e
`G-Grams`	$gf(e)$	n-gram count of concept e
`MSN Query Log`	$qe\_cnt(e)$	Number of exact matches of e in the query log
	$qp\_cnt(e)$	Number of times e occurs within the query log
`Wikipedia Titles`	$we\_cnt(e)$	Does a concept e appear as a Wikipedia title?
	$wp\_cnt(e)$	Number of times e occurs in a Wikipedia title.

including the so-called full dependence assumption (Metzler and Croft 2005) and other models that focus on key dependencies in the queries (Bendersky et al. 2009).

4 Concept Importance Features

In this section, we describe the features used for determining the importance of a term or a bigram in a weighted sequential dependence model. Recall that parameters $\lambda(q_i)$ and $\lambda(q_i, q_{i+1})$, which determine the concept weights, are represented as a weighted linear combination of features $g^u(q_i)$ and $g^b(q_i, q_{i+1})$. These features are defined over concepts (either terms or bigrams) and are independent of a specific document. This fact allows us to combine the statistics of the underlying document corpus with the statistics of various external data sources to achieve a potentially more accurate weighting. Accordingly, we divide the features used for concept importance weighting into two main types, based on the type of information they are using.

The first type, the *endogenous*, or collection-dependent, features are akin to standard weights used in information retrieval. They are based on collection frequency counts and document frequency counts calculated over a particular document corpus on which the retrieval is performed.

The second type, the *exogenous*, or collection-independent, features are calculated over an array of external data sources. The use of such sources was found to be beneficial for information retrieval models in previous work (Bai et al. 2008; Bendersky and Croft 2008; Lease et al. 2009). Some of these data sources provide better coverage of terms, and can be used for smoothing sparse concept frequencies calculated over smaller document collections. Others provide more focused sources of information for determining concept importance. We consider three external data sources: (i) a large collection of Web n-grams, (ii) a sample of a query log, and (iii) Wikipedia. Although there are numerous additional data sources that could be potentially used, we intentionally limit our attention to these three sources as they

Table 5.3 Summary of language modeling-based unigram and concept weighting functions. Here, $tf_{e,D}$ is the number of times concept e matches in document D, $cf_{e,D}$ is the number of times concept e matches in the entire collection, $|D|$ is the length of document D, and $|C|$ is the total length of the collection. Finally, μ is a weighting function hyperparameter that is set to 2500

Weighting
$f_T(q_i, D) = \log\left[\frac{tf_{q_i,D}+\mu\frac{cf_{q_i}}{\lvert C\rvert}}{\lvert D\rvert+\mu}\right]$
$f_O(q_i, q_{i+1}, D) = \log\left[\frac{tf_{\#1(q_i,q_{i+1}),D}+\mu\frac{cf_{\#1(q_i,q_{i+1})}}{\lvert C\rvert}}{\lvert D\rvert+\mu}\right]$
$f_U(q_i, q_{i+1}, D) = \log\left[\frac{tf_{\#uw8(q_i,q_{i+1}),D}+\mu\frac{cf_{\#uw8(q_i,q_{i+1})}}{\lvert C\rvert}}{\lvert D\rvert+\mu}\right]$

are available for research purposes, and can be easily used to reproduce the reported results.

The first source, *Google n-grams* corpus[1], contains the frequency counts of English n-grams generated from approximately 1 trillion word tokens of text from publicly accessible Web pages. We expect these counts to provide a more accurate frequency estimator, especially for smaller corpora, where some concept frequencies may be underestimated due to the collection size.

In addition, we use a large sample of a query log consisting of approximately 15 million queries[2]. We use this data source to estimate how often a concept occurs in user queries. Intuitively, we assume a positive correlation between an importance of a concept for retrieval and the frequency with which it occurs in queries formulated by search engine users.

Finally, the third external data source is a snapshot of Wikipedia article titles[3]. Due to the large volume and the high diversity of topics covered by Wikipedia (∼3 million articles in English alone), we assume that important concepts will often appear as (a part of) article titles in Wikipedia.

Table 5.2 details the statistics used for determining concept weights. As described above, these statistics are based either on the collection or on one of the external data sources. These statistics are used to compute term and bigram features ($g^u(q_i)$ and $g^b(q_i, q_{i+1})$, respectively) in the weighted sequential dependence model (see Eq. 5.5).

For computing the term features, we calculate the statistics presented in the Table 5.2 for all query terms q_i. This provides us with 7 features $g^u(q_i)$ for determining term importance weights.

To compute the bigram features, we calculate the statistics presented in the Table 5.2 for all sequential query term pairs q_i, q_{i+1}. For computing collection statistics, we use both the "exact phrase" matches and "unordered window" matches,

[1] Available from the Linguistic Data Consortium catalog.

[2] Available as a part of Microsoft 2006 RFP dataset.

[3] Available at: http://download.wikimedia.org/enwiki/.

as described in Table 5.3. In addition, as bigram features, we compute a ratio $\frac{s(q_i q_{i+1})}{s(q_i)s(q_i+1)}$ for every statistic s in the Table 5.2. Overall, the combination of the above statistics, provides us with 18 features $g^b(q_i, q_{i+1})$ for determining bigram importance weights.

For each of the data sources, we use a number of standard functions to calculate the features. For endogenous features, we use collection frequency (*cf*) and document frequency (*df*). We calculate these functions for both unigram, exact bigram matches ($\#ow1(b_1, b_2)$) and unordered bigram matches within a window of fixed size ($\#uw8(b_1, b_2)$). For exogenous features, we use exact n-gram matches ($e\_cnt(\cdot)$) and partial n-gram matches ($p\_cnt(\cdot)$) (note that by definition $p\_cnt(\cdot) \geq e\_cnt(\cdot)$, and for the Wikipedia titles $e\_cnt(\cdot)$ is a binary feature).

5 Evaluation

This section presents the experimental results of the method just described. We start by detailing the experimental set-up in Sect. 5.1. Next, in Sect. 5.2, we perform a comprehensive evaluation of the method using several publicly available corpora used at the Text REtrieval Conference (TREC), including newswire and Web collections. Finally, to illustrate the benefits of the approach for Web search, in Sect. 5.3 we test the performance of the method using a proprietary Web corpus and a large sample of user queries.

5.1 Experimental Setup

The retrieval experiments are set up as follows. For all TREC collections, we obtain an initial list of top-1000 results retrieved by an unweighted sequential dependence model. This initial ranking provides a very competitive baseline, as the sequential dependence model was consistently shown to outperform the standard bag-of-word models (Lease 2009; Metzler and Croft 2005). We append all the non-retrieved relevant documents to the top-1000 list, and use this set of results for training and evaluating all the compared retrieval models.

For the proprietary Web corpus, we only index Web pages that have relevance judgments for the query samples. Training and evaluation of the retrieval models is done using this set of judged Web pages, which is a common evaluation practice for this type of test collection. There are, on average, 27 judgments per query.

We compare the performance of the weighted sequential dependence model (`WSD`) to two baseline retrieval models. The first is the query-likelihood model (Ponte and Croft 1998) (`QL`), a standard bag-of-words retrieval model implemented by the Indri search engine. The second is the unweighted sequential dependence model (`SD`). All the initial retrieval parameters are set to default Indri values, which reflect the best-practice settings. All the training/evaluation is done using five fold

cross-validation. The statistical significance of the differences in the performance is determined using a two-sided Wilcoxon sign test, with $\alpha < 0.05$.

We measure the performance using standard retrieval metrics for TREC and Web corpora. For TREC corpora, which uses binary relevance judgments, we use precision at top 10 documents retrieved (P@10) and mean average precision (MAP) at rank 1000. See Appendix B for a more detailed description of these measures. When estimating the parameters for the `WSD` model, we directly optimize MAP (see the next chapter for further details).

For the Web corpus, which uses graded relevance judgments, we use the discounted cumulative gain measure (DCG) (Järvelin and Kekäläinen 2002) at ranks 5 and at the total depth of the ranked list. Relevance is judged as either Perfect, Excellent, Good, Fair, or Bad. The corresponding DCG gains for these grades are 10, 7, 3, 0.5, and 0, respectively. In the direct optimization of the weighted dependence model, we use normalized DCG as the target metric.

5.2 TREC Evaluation

In this section, we describe the retrieval results obtained by the model on three standard TREC collections. A summary of the corpora used for these experiments is shown in Appendix A. We note that collections vary both by type (ROBUST04 is a newswire collection, while W10g and GOV2 are Web collections), number of documents and number of available topics, thus providing a diverse experimental set-up for assessing the robustness of the weighted dependence model.

In the evaluation we use both the *title* and the *description* portions of TREC topics as queries. *Title* queries are generally short, and can be viewed as a keyword queries on the topic. *Description* queries are generally more verbose and syntactically richer natural language expressions of the topic. For instance queries *pet therapy* and *How are pets or animals used in therapy for humans and what are the benefits?* are examples of title and description queries on the same topic, respectively.

5.2.1 Retrieval Results

Table 5.4 shows the summary of the retrieval results for the three TREC collections on both *title* and *description* queries. It is evident that both sequential dependence models (`SD` and `WSD`) significantly outperform the query-likelihood model `QL` in almost all the cases on all the metrics. This verifies the positive impact of term dependencies on the retrieval performance.

From the two sequential dependence models, weighted sequential dependence model (`WSD`) significantly outperforms the unweighted one (`SD`) on all collections in terms of MAP (which is used as the metric for direct optimization). The gains in MAP range between 1.6% and 19.5%, and are statistically significant for all collections and both query types.

Table 5.4 Comparison of retrieval results for *title* (top table) and *description* (bottom table) TREC queries with query-likelihood (QL), sequential dependence model (SD) and weighted sequential dependence model (WSD). Numbers in parentheses indicate % improvement in MAP over QL/SD (if available)

title	ROBUST04		W10g		GOV2	
	P@10	MAP	P@10	MAP	P@10	MAP
QL	0.4225	0.2493	0.2560	0.1904	0.5342	0.3019
SD	0.4410*	0.2661*	**0.2890***	0.2063*	**0.5785***	0.3247*
WSD	**0.4462***	**0.2721*†**	**0.2890***	**0.2220*†**	0.5779*	**0.3338*†**

desc	ROBUST04		W10g		GOV2	
	P@10	MAP	P@10	MAP	P@10	MAP
QL	**0.4269**	0.2507	0.3270	0.1971	0.5168	0.2606
SD	0.4177	0.2558	0.3610*	0.2032	**0.5356**	0.2694*
WSD	**0.4269**	**0.2718*†**	**0.3710***	**0.2523*†**	0.5181	**0.2738*†**

∗ Statistically significant difference with QL
† Statistically significant difference with SD

It is interesting to note that even on P@10, which was not directly optimized for, WSD is more effective than SD in all but two comparisons (P@10 for GOV2). The gains observed are as high as 2.7% for P@10. We expect that even higher gains for P@10 can be attained by WSD by directly training the model for these measures of interest rather than MAP.

5.2.2 Feature Analysis

In this section we perform a detailed feature analysis, in order to identify the key elements in the success of the weighted sequential model, as compared to its unweighted counterpart.

Unigrams and Bigrams Table 5.5 compares the impact on the retrieval effectiveness of the importance weights assigned by WSD to either unigrams or bigrams in the sequential dependence model. Recall that the weighted sequential dependence model WSD is derived from its unweighted counterpart by replacing the static sequential dependence parameters λ_T, λ, and λ_U with concept dependent parameters $\lambda(q_i)$ and $\lambda(q_i, q_{i+1})$, as shown in Eq. 5.5.

The WSD-UNI model, shown in Table 5.5, is obtained by replacing λ_T with the term dependent $\lambda(q_i)$, while fixing the values of λ_O and λ_U to those of the unweighted sequential dependence model. Alternatively, WSD-BI model is obtained by replacing λ_O and λ_U with the term dependent $\lambda(q_i, q_{i+1})$, while fixing the value of λ_T.

Table 5.5 Comparison of retrieval results for *title* (left) and *description* (right) TREC queries with either only unigram features (WSD-UNI), only bigram features (WSD-BI) or both

title	ROBUST04	W10g	GOV2	*desc*	ROBUST04	W10g	GOV2
WSD	**0.2721**	**0.2220**	0.3338	WSD	**0.2718**	**0.2523**	**0.2738**
WSD-UNI	0.2685†	0.2188	**0.3343**	WSD-UNI	0.2717	0.2486	0.2677†
WSD-BI	0.2675†	0.2065†	0.3258	WSD-BI	0.2602†	0.2043†	0.2700

† Statistically significant difference with WSD

Table 5.5 compares the performance of both WSD-UNI and WSD-BI models to the performance of the fully weighted sequential dependence model (WSD). We note that while, in general, both WSD-UNI and WSD-BI outperform SD, in most cases WSD-UNI outperforms WSD-BI as well. This indicates that a unigram weighting has more impact on the retrieval performance than the bigram weighting. This result is in line with previous results reported by Lease for TREC collections (Lease 2009), which showed that by solely weighting unigrams, one can significantly outperform the unweighted sequential model baseline.

Another important finding shown in Table 5.5, is that WSD, which combines both unigram and bigram weights, outperforms WSD-UNI in 5 out of 6 comparisons, and always outperforms WSD-BI. In addition, WSD attains statistically significant differences in comparison with WSD-UNI for *description* queries on a large Web collection GOV2. This fact underscores the importance of weighting for all the concepts in the sequential dependence model.

Endogeneous and Exogenous Features Recall from Sect. 4 that WSD uses two types of features for estimating concept importance: endogeneous (collection-dependent) and exogeneous (collection-independent). While applying collection-dependent features for term weighting has been extensively studied in traditional information retrieval (Salton and Buckley 1988), the research on combining them with external sources of information is more recent (Bendersky and Croft 2008; Lease et al. 2009). Therefore, it is interesting to examine the contribution of each of these feature types to the overall model performance.

Table 5.6 compares the performance of the weighted sequential dependence model when either only endogeneous (WSD-ENDO) or only exogeneous (WSD-EXO) features are used to the performance of the fully weighted sequential dependence model (WSD). It is evident from Table 5.6 that using either the endogeneous or the exogeneous features results in comparable performance, and both of them outperform the unweighted dependence model. This indicates that both of these features are useful for learning the optimal weights for WSD. In both cases, however, their combination results in gains in MAP in 5 out of 6 cases. In addition, we found that both WSD-ENDO and WSD-EXO display statistically significant differences with WSD on a large Web collection GOV2 for *description* queries.

Table 5.6 Comparison of retrieval results for *title* (left) and *description* (right) TREC queries with either only endogenous features (WSD-ENDO), only exogenous features (WSD-EXO) or both

title	ROBUST04	W10g	GOV2	*desc*	ROBUST04	W10g	GOV2
WSD	**0.2721**	**0.2220**	0.3338	WSD	0.2718	**0.2523**	**0.2738**
WSD-ENDO	0.2685†	0.2176	0.3281	WSD-ENDO	0.2707	0.2328	0.2695†
WSD-EXO	0.2701	0.2119†	**0.3354**	WSD-EXO	**0.2733**	0.2468	0.2733†

† Statistically significant difference with WSD

Table 5.7 Comparison of retrieval results over a sample of Web queries with query-likelihood (QL), sequential dependence model (SD) and weighted sequential dependence model (WSD). Numbers in parentheses indicate % improvement in DCG over QL/SD (if available)

	Len-2		Len-3		Len-4+	
	DCG@5	DCG	DCG@5	DCG	DCG@5	DCG
QL	2.231	10.750	2.290	8.204	1.691	5.844
SD	2.733	11.539	**2.971**	**9.139**	2.383	6.681
WSD	**2.754**	**11.585**	2.929	9.087	**2.443**	**6.741**

All the differences are statistically significant

5.3 Large-Scale Web Evaluation

Previous research has shown that modeling sequential term dependencies has a significant positive impact on retrieval performance in the Web search setting (Bai et al. 2008; Metzler and Croft 2005; Mishne and de Rijke 2005). Given the retrieval performance gains obtained from using the weighted variant of the sequential dependence model demonstrated on TREC collections in the previous section, the following set of experiments explores whether these gains can be directly transferred into a Web search setting. To this end, in this section we test the ranking with a weighted sequential dependence model on a proprietary Web corpus provided by a large commercial search engine.

To differentiate between the effect of concept weighting on queries of varying length, as was done in the case of TREC corpora, we divide the queries into three groups based on their length. Length is defined as a number of word tokens separated by space in the query. The first group of queries (*Len-2*) includes very short queries of length two. The second group (*Len-3*) includes queries of length three. The third group (*Len-4+*) consists of more verbose queries of length varying between four and twelve. While the queries in the first two groups mostly have a navigational intent, the queries in the third group tend to be more complex informational queries. For each group, we randomly sample 1,000 Web search queries for which relevance judgments are available. We then train and evaluate (using five fold cross-validation) a separate sequential dependence model and weighted sequential dependence model for each group.

Table 5.8 Concept weights generated for query "information about peer gynt suite"

Concept	Weight
information	0.0210
about	0.0193
peer	0.1591
gynt	0.4114
suite	0.1555
information about	−0.0399
about peer	0.0122
peer gynt	0.0891
gynt suite	0.0399

5.3.1 Retrieval Results

Table 5.7 shows the summary of the retrieval results on the three query groups. To demonstrate the impact on the relevance at the top ranks of the retrieved list we report the DCG@5. To demonstrate the overall ranking quality, we report the results for DCG at unlimited depth (denoted DCG).

Table 5.7 demonstrates two important findings. First, including term dependence information is highly beneficial for queries of all lengths. `SD` attains up to 12.5% improvement over `QL`, which is a bag-of-words model. This result is highly significant, given the large size of the query set. Second, concept weighting results in significant improvements for longer (*Len-4+*) queries, and its performance is comparable for shorter queries to the performance of the unweighted dependence model (slight improvement on *Len-2* and slight decrease in performance on *Len-3*). For group *Len-4+*, `WSD` attains improvement of close to 2.5% for DCG@5, a highly significant improvement, especially when taking into account the importance of relevance at top ranks for the Web search task.

5.3.2 Feature Analysis

Similarly to the feature analysis performed in Sect. 5.2.2 for TREC corpora, in this section we analyze the importance of different weights and features in the weighted sequential model for the Web corpus.

Unigrams and Bigrams Table 5.9 compares the impact on the retrieval effectiveness of the importance weights assigned by `WSD` to either unigrams or bigrams in the sequential dependence model. Notice that, contradictory to what was observed in Table 5.5 for the TREC data, the bigram weights have more impact on the retrieval effectiveness than the unigram weights. For short queries in groups *Len-2* and *Len-3*, using bigram weights alone and omitting the unigram weights results in a slightly higher DCG at all measured ranks than using the fully weighted dependence model.

Table 5.9 Comparison of retrieval results for a sample of Web queries with either only unigram features (WSD-UNI), only bigram features (WSD-BI) or both

	Len-2		Len-3		Len-4+	
	DCG@5	DCG	DCG@5	DCG	DCG@5	DCG
WSD	2.754	11.585	2.929	9.087	**2.443**	**6.741**
WSD-UNI	2.743	11.556	2.963	9.132	2.379	6.677
WSD-BI	**2.758**	**11.602**	**2.967**	**9.132**	2.409	6.711

All the differences are statistically significant

Table 5.10 Comparison of retrieval results for a sample of Web queries with either only endogenous features (WSD-ENDO), only exogenous features (WSD-EXO) or both

	Len-2		Len-3		Len-4+	
	DCG@5	DCG	DCG@5	DCG	DCG@5	DCG
WSD	**2.754**	**11.585**	**2.929**	**9.087**	2.443	6.741
WSD-ENDO	2.687	11.487	2.924	9.085	**2.455**	**6.760**
WSD-EXO	2.749	11.575	2.919	9.079	2.439	6.732

All the differences are statistically significant

A likely explanation for this effect is the dominance of navigational intent for short queries in Web search. TREC topics, including the short *title* queries, mostly have an informational intent and often consist of several separate concepts of unequal importance (e.g., "abandoned mine reclamation"). Short two-three word Web queries, on the other hand, often consist of a single navigational bigram ("yahoo mail"), or a bigram followed by an auxiliary term ("yahoo mail login").

Compared to the first two groups, using both unigram and bigram weights in queries in group *Len-4+* results in a better performance than using either of them alone, which is in line with the results for the TREC collections. We hypothesize that this stems from the fact that a higher percentage of these queries have an informational intent, and they contain both unigram and bigram concepts of varying importance ("best metal songs of the 1980s").

Overall, as evident from Table 5.9, the impact of concept weights is influenced both by the query type and by the collection. While the weighted sequential model can naturally incorporate weighted and unweighted concepts, the optimal weighting policy has to be determined using training on the available data.

Endogenous and Exogenous Features Table 5.10 compares the performance of the weighted sequential dependence model when either only endogeneous (WSD-ENDO) or only exogeneous (WSD-EXO) features are used to the performance of the fully weighted sequential dependence model (WSD). It is evident from Table 5.10 that using either the endogeneous or the exogeneous features results in most cases

in comparable performance. Similarly to WSD, both of them outperform the unweighted dependence model on queries in group *Len-4+*.

For shorter queries in the first two groups combining the two types of features results in a better performance than using either one in isolation. For queries in a group *Len-4+* using endogenous features alone results in a slightly better performance than the WSD, however the difference is relatively minor (0.3% improvement of the DCG metric). In addition, the impact of exogeneous features on the overall retrieval performance of the Web queries might be potentially boosted by including additional external sources, instead of just three, as is currently done. For instance, a larger and a more recent sample of user queries than the one used in this study could be employed.

As a general "rule of thumb" strategy, a combination of both endogenous and exogenous features appears to be the preferred option both for the TREC and for the Web corpora.

Some illustrative examples of learned concept weights are shown in Table 5.8.

Chapter 6
Model Learning

1 Parameter Estimation

In this section we describe how to estimate the parameters of linear feature-based models, which corresponds to solving the optimization proposed earlier in this chapter. This is often called the *learning-to-rank* problem. The topic has recently gained popularity in the machine learning and information retrieval fields, generating a large number of tools and techniques. While many of these techniques take a purely machine learning approach to the problem, we attempt to focus more on the important information retrieval aspects of the problem. We believe this is important, because critical information retrieval issues are often ignored during the development of new learning-to-rank techniques. This is somewhat unfortunate, since valuable insights can often be gleaned by attacking these types of problems from an information retrieval perspective (e.g., via feature engineering, failure analysis, etc.) instead of a machine learning perspective.

Many of the linear feature-based models that have been used for information retrieval use parameters that are estimated using maximum likelihood, maximum *a posteriori*, or maximum margin techniques. These techniques, however, do not maximize the correct metric. Classification accuracy, likelihood, and margin size are generally of little concern when ranking documents. It may be argued that estimating parameters by maximizing the likelihood of some training data or minimizing classification error is optimizing a function that is correlated with the underlying retrieval metric, such as mean average precision. However, this has been shown to be experimentally invalid, and it can also be shown to be theoretically invalid, as well. This phenomenon, where the metric being optimized diverges from the actual metric of interest, is known as *metric divergence* (Morgan et al. 2004). Hence, the appropriate way to estimate the parameters of linear feature-based models is to optimize the model parameters with respect to some rank-based information retrieval metric.

The remainder of this section describes various approaches for solving this optimization problem, including two algorithms that perform a direct search over the multinomial manifold. These algorithms can easily handle large training sets, such

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8_6,

as those that typically arise in information retrieval. They can also handle the highly unbalanced nature of the training data. Although these techniques have nice properties, they also have several pitfalls, as we will show. We will also briefly describe other learning-to-rank approaches that have been proposed recently.

1.1 Direct Search

We now describe two direct search techniques that operate on the multinomial manifold. Here, direct search refers to the fact that the optimization problem in solved in the original metric space. It is important to remember that these metric spaces are typically non-smooth with respect to the parameter settings, which prevents us from applying standard, gradient-based optimization techniques, such as gradient ascent. The techniques described here are very simple, yet have been shown to be very effective.

1.1.1 Grid Search

The most naïve approach to solving the optimization problem is to perform an exhaustive grid search over the parameter space. That is, we place a grid over the parameter space and evaluate $E(\mathcal{R}_\Lambda; \mathcal{T})$ at every grid intersection, outputting the parameter setting that yields the maximum at the end.

A grid search over $\mathbb{R}^d$ is unbounded and ill-defined. For this reason, we restrict the discussion to the case where the parameter space is the multinomial manifold. For this case, the grid search is bounded and can be easily implemented.

Given a parameter $\epsilon = \frac{1}{K}$ for $K \in \mathbb{Z}^+$ that controls how fine grained the grid is, we define $\mathcal{G}$, the set of grid points in $\mathbb{P}^{d+1}$ that we search over, as

$$\begin{aligned}\mathcal{G} &= \left\{ \Lambda = (k_1\epsilon \dots k_d\epsilon) : \sum_i k_i\epsilon = 1, k_i \in \mathbb{N} \right\} \\ &= \left\{ \Lambda = (k_1\epsilon \dots k_d\epsilon) : \sum_i k_i = K, k_i \in \mathbb{N} \right\}. \end{aligned} \tag{6.1}$$

It is clear from this definition that $|\mathcal{G}|$, the number of parameter values we must evaluate E at, depends both on d (the number of parameters) and K (how fine grained the grid is). In fact, $|\mathcal{G}| \in \Theta(K^{d-1})$. Therefore, a grid search is feasible only if both d and K are relatively small. For larger values, other training methods that do not require as many sorting operations and metric evaluations must be used. Although the grid search algorithm is relatively costly, it is guaranteed to find a global maximum as K gets large. Algorithm 1 provides a simple implementation of the grid search algorithm.

Algorithm 1 Grid Search

$(k_1, k_2, \ldots, k_d) \leftarrow (0, 0, \ldots, 0)$
$\Lambda^* \leftarrow \{\}$
$E^* \leftarrow -\infty$
while $k_d \leq K$ **do**
  $i \leftarrow 1$
  $k_i \leftarrow k_i + 1$
  while $i < d$ and $k_i = K$ **do**
    $k_i \leftarrow 0$
    $i \leftarrow i + 1$
    $k_i \leftarrow k_i + 1$
  end while
  if $\sum_{i'} k_{i'} = K$ **then**
    $\Lambda \leftarrow (\frac{k_1}{K}, \frac{k_2}{K}, \ldots, \frac{k_d}{K})$
    $E \leftarrow E(\mathcal{R}_\Lambda; \mathcal{T})$
    if $E > E^*$ **then**
      $E^* \leftarrow E$
      $\Lambda^* \leftarrow \Lambda$
    end if
  end if
end while
Return Λ^*

1.1.2 Coordinate Ascent

Coordinate ascent is another technique that can be used to solve non-smooth optimization problems. One of the benefits of the algorithm is that it reduces multivariate search problems into a set of single variable problems, which are often easier to tackle in non-smooth spaces. The algorithm first chooses one parameter to be free. It then holds all other parameters as fixed and optimizes the objective function over the single free parameter. This produces an uphill step along one coordinate dimension. This process is repeated for all parameters over a number of iterations. The technique is known to converge slowly on objective functions with long ridges. Variations of the method, including Powell's method, have been proposed to overcome this issue (Press et al. 1992).

Coordinate ascent can be applied to the optimization problem under consideration regardless of whether we choose to optimize in the original Euclidean parameter space ($\mathbb{R}^d$) or the mapped multinomial parameter space ($\mathbb{P}^{d-1}$). Optimizing over the manifold may be beneficial due to the reduction in the number of repeated local extrema.

If coordinate ascent is performed over the multinomial manifold, then only a minor modification to the original algorithm is necessary. All one dimensional searches done by the algorithm will be performed as if they were being done in $\mathbb{R}^d$. However, this does not ensure that the updated parameter estimate will be a point on the

Algorithm 2 Coordinate Ascent

Initialize $\Lambda^0 \leftarrow (\lambda_1^0, \lambda_1^0, \ldots, \lambda_d^0)$
$t \leftarrow 1$
repeat
  $\Lambda^t \leftarrow \Lambda^{t-1}$
  for i from 1 to d **do**
    $\lambda_i^t \leftarrow \arg\max_{\lambda_i} E(\mathcal{R}_{\Lambda^t}; \mathcal{T})$
    $\lambda_j^t \leftarrow \frac{\lambda_j^t}{\sum_i' \lambda_{i'}^t} \; \forall j$ (optional)
  end for
until $|E(\mathcal{R}_{\Lambda^t}; \mathcal{T}) - E(\mathcal{R}_{\Lambda^{t-1}}; \mathcal{T})| > \epsilon$
Return Λ^t

manifold. Therefore, after a step is taken in $\mathbb{R}^d$, we project the point back onto the manifold, which we showed is always possible. Note that this projection preserves the function value since the unnormalized and projected parameter estimates lead to equivalent rankings. Therefore, the optimization is implicitly being done in a space that we know how to optimize over ($\mathbb{R}^d$), but is continually being projected back onto to the manifold.

The coordinate ascent algorithm is given in Algorithm 2. In the algorithm, Λ^t denotes the parameter setting during iteration t. The algorithm first initializes Λ^0, which can be done uniformly (i.e., $\lambda_i^0 = \frac{1}{d}$), randomly, or in a more informed manner using prior knowledge about the importance of each feature. Then, each λ_i^t is updated according to $\arg\max_{\lambda_i} E(\mathcal{R}_{\Lambda^t}; \mathcal{T})$, which holds all parameter values fixed except λ_i, and finds the setting of λ_i that results in the maximum evaluation metric score. This single dimensional search problem can be solved using a line search. The line search can be done exhaustively or finite difference derivatives can be estimated and used in lieu of exact derivatives. Of course, if E is partially differentiable with respect to each parameter, then it may be possible to solve the single dimensional search problem analytically. After λ_i^t is updated, the updated parameter vector Λ^t is optionally projected back onto the multinomial manifold.

The algorithm continues until the evaluation metric does not change more than ϵ between subsequent iterations. If ϵ is set to 0, then the algorithm stops only when no more uphill moves are possible. Setting $\epsilon > 0$ may cause the algorithm to return a solution that is not maximal. However, it also results in faster training and can help minimize overfitting.

Our implementation of the algorithm uses a line search to solve the single dimensional search problem, projects updated parameters onto the multinomial manifold, and uses $\epsilon = 0.0001$. After extensive evaluation, it was found that these settings resulted in the most well-behaved configuration of the algorithm.

1.1.3 Discussion

Finding the maximum of an arbitrary evaluation function E using direct search can be very difficult and error-prone, especially in high-dimensional space. Only a grid search method, with a suitably chosen granularity, is guaranteed to find a global maxima. Coordinate ascent is a local search technique that only finds a global maxima if the evaluation function E is concave. The experiments using this approach, show that, for a certain set of term and phrase features, mean average precision is approximately concave over a wide range of collections. This may be the case for many related applications and feature sets, but is not true in general, as was pointed out in Gao et al. (2005). For functions with many local maxima, a multiple random restart strategy can be used to increase the chances of finding a global solution.

Another potential disadvantage of the direct search techniques presented here is the fact that they are fully supervised. If few or no training data exist, then unsupervised and active learning (Shen and Zhai 2005) techniques from machine learning can potentially be employed. However, such methods are out of the scope of the current work.

Despite these disadvantages, the approach has the advantage that it can make use of all of the training data and does not suffer in the face of highly unbalanced training data. When training using maximum likelihood or SVMs, it is often important to have balanced training data. However, in information retrieval it is very often the case that there are many more relevant documents compared to non-relevant documents for a given query. For this reason, the training data are very unbalanced. Nallapati found that the data needed to be balanced in order to achieve good generalization performance (Nallapati 2004). Balancing was done by undersampling the majority (non-relevant) class. Although this led to improved performance over the unbalanced case, it had the negative effect of throwing away valuable training data. Other solutions to the unbalanced data problem for SVMs exist that do not require training data to be compromised, such as allowing separate costs for training errors in the positive and negative classes (Morik et al. 1999). Since the coordinate ascent approach does not make any implicit or explicit assumptions about the underlying distribution, we can use the training data in their entirety.

1.2 Optimization Using Surrogate Functions

A number of other estimation techniques have been proposed for training linear feature-based models for information retrieval. The common thread among these approaches is that they do not attempt to directly search in the evaluation metric space. Instead, they typically optimize some *smooth* surrogate function that is related to the evaluation metric space. By using a smooth, typically convex, surrogate function, it is possible to apply standard optimization machinery to the problem. However, it is not always the case that the optimum of the surrogate is equal to the optimum of the actual metric. For example, the surrogate function may be a lower

or upper bound on the metric. Although these surrogates may yield reasonable estimates, many exhibit metric divergence. In certain cases, however, the optimum of the surrogate is equal to the optimum of the evaluation metric, thereby eliminating any metric divergence. In the remainder of this section we summarize the various techniques proposed and describe the properties of their surrogate functions.

1.2.1 Perceptron Learning

Gao et al. proposed using a perceptron-based algorithm to optimize mean average precision (Gao et al. 2005). The technique uses pairwise preferences as training data (Wong and Yao 1988). That is, training data, denoted by $\mathcal{R}$, come in the form of tuples of the form $(d_i, d_j)_q$, which indicates that document i should be ranked higher than document j for query q. There are various ways to derive such preferences from manual relevance judgments or click-through data (Joachims 2002). The perceptron learning algorithm is demonstrated in Algorithm 3. The perceptron

Algorithm 3 Perceptron Learning

```
Initialize Λ^0 ← (1, 0, ..., 0)
for t from 1 to MAX_ITERATIONS do
    for each (d_i, d_j)_q ∈ R do
        if Λ^T f(d_j, q) > Λ^T f(d_i, q) then
            λ_i ← λ_i + η(f(d_i, q) − f(d_j, q))
        end if
    end for
end for
Return Λ
```

approach, like the coordinate ascent algorithm described earlier, is not guaranteed to find a global maxima. Instead, the perceptron learning algorithm optimizes a lower bound on mean average precision. Therefore, metric divergence may be a problem. In addition, this technique cannot easily be applied to other evaluation metrics. Despite this, the algorithm has been shown to produce reasonable effectiveness over a wide range of data sets.

1.2.2 RankNet

Another approach, based on neural networks, called RankNet, has recently been proposed (Burges et al. 2005). A RankNet is trained using gradient descent over the following differentiable cost function:

$$C(\mathcal{Q}, \mathcal{R}) = \sum_{q \in \mathcal{Q}} \sum_{(i,j) \in \mathcal{R}} -\hat{P}_{q,i,j} \log P_{q,i,j} - (1 - \hat{P}_{q,i,j}) \log(1 - P_{q,i,j}), \qquad (6.2)$$

where $\mathcal{Q}$ is the set of queries, $\mathcal{R}$ is the set of pairwise preferences used for training, $P_{q,i,j} = \frac{1}{1+\exp[s(q,d_j)-s(q,d_i)]}$ is the probability that document i ranks higher than document j for query q under the current neural network ($s(q, d_i)$ is the score of document d for query q as computed by the neural network), and $\hat{P}_{q,i,j}$ is the target probability of document i being ranked higher than j (obtained from training data).

The gradients of this smooth cost function can be computed analytically, which makes it easy to apply gradient descent to solve the optimization problem. However, the model suffers from standard neural network training issues, such as local minima. In addition, the cost function, which is simply the cross entropy between the target distribution and the distribution modeled using the neural network, does not minimize (or maximize) any specific retrieval metric. Therefore, it is vulnerable to metric divergence. Several recent studies have attempted to address this problem (Burges et al. 2007; Matveeva et al. 2006).

We also note that RankNet is a linear feature-based model when the underlying neural network has no hidden layers. The underlying ranking function, in the presence of hidden layers, may be highly non-linear.

1.2.3 Support Vector Machine Optimization

Finally, Joachims proposed a large margin training technique for multivariate performance measures (Joachims 2002, 2005). The technique uses a surrogate objective function based on SVM using structured outputs that can be solved using quadratic programming. The approach can maximize a variety of information retrieval metrics, such as precision at k, precision-recall break-even, and area under the ROC curve. In fact, any metric that can be computed based solely on a contingency table can be maximized efficiently. One particularly nice property of these metrics is that the maximum found is the actual maximum, and therefore, there is no metric divergence. However, this is not the case for any arbitrary metric. Recently, approaches based on this work have been proposed to optimize lower bounds of average precision (Yue et al. 2007) and nDCG (Le and Smola 2007). Furthermore, since the method is based on SVMs, it is easy to employ the "kernel trick" and implicitly project inputs into a higher, possibly infinite, dimensional space.

1.2.4 Discussion

There are two downsides to these types of approaches. First, they specifically work for one metric or a family of metrics. Second, many of them suffer from metric divergence, even though the resulting optimization problems are easier and more efficient to solve using well-established optimization techniques. The grid search and coordinate ascent algorithms, however, do not suffer from either of these problem.

There currently is no well developed understanding of best practices for estimating the parameters of linear feature-based information retrieval models. Most studies have looked at traditional machine learning problems, which typically differ

from information retrieval tasks. Therefore, an interesting, and necessary, direction of future work is to undertake a comprehensive evaluation of these techniques, in terms of how effective they are across a wide range of retrieval data sets and metrics, how well they generalize, and how efficient they are. Another critical question that must be answered is whether or not non-linear models, such as those that arise when using neural networks or SVMs with non-linear kernels, are more effective than simple linear models.

2 Feature Selection

As we showed in Chap. 2, many different types of retrieval models have been proposed throughout the years. These include Boolean, vector space, logic-based, probabilistic, and feature-based. One critical factor that must be considered when developing information retrieval models is the type of features to be used or modeled. Term frequency, inverse document frequency, document length, and term proximity are the fundamental features that are used in most of the modern information retrieval models including BM25 (Robertson and Walker 1994), language modeling (Song and Croft 1999), divergence from randomness (DFR) (Amati and van Rijsbergen 2002), axiomatic approach to IR (Fang and Zhai 2005), and the MRF model.

However, most of these models make use of hand selected, statistically inspired, or implicit features. Therefore, it is often difficult to adapt these types of models to new tasks, especially when the task has new, completely different types of features associated with it. Applying these models to new tasks typically requires an information retrieval expert to modify the underlying model in some way in order to properly account for the new types of features. This is a common theme in information retrieval modeling. Examples include incorporating PageRank as a prior into the BM25 model (Craswell et al. 2005b), allowing term proximity information as evidence in BM25 (Büttcher et al. 2006a), modeling document structure in both language modeling and BM25 (Ogilvie and Callan 2003; Robertson et al. 2004), including term dependence in the DFR model (Peng et al. 2007), and allowing term associations in the axiomatic model (Fang and Zhai 2006). These examples illustrate that incorporating new types of evidence and features into existing retrieval models is often non-trivial and can require significant amounts of human involvement.

Therefore, it is desirable for models to be flexible and robust enough to easily handle a wide range of features and provide a mechanism for automatically selecting relevant features. Then, given a large pool of candidate features, it would be possible to automatically learn the best model. Under this model learning paradigm, there would no longer be a need to manually tune or modify some existing retrieval model whenever a new task or data set is encountered. Instead, attention could be paid to developing a rich pool of features that are widely applicable.

We argue that the MRF model, and similar types of models, such as Gao et al.'s linear discriminant model (Gao et al. 2005), are the correct types of models to use when model flexibility and robustness are important. In this chapter, we describe an

automatic, supervised feature selection algorithm that can be used in conjunction with these types of models. The algorithm is general and can be applied to a wide range of feature sets, evaluation metrics, and methods for learning to rank. Besides being more robust and flexible, we also show that models constructed using the algorithm are often significantly more effective than the hand built basic MRF models described in Chap. 3.

2.1 Related Work

A number of feature selection techniques for random field models have been proposed in the machine learning literature (Della Pietra et al. 1997; McCallum 2003). The algorithm is an adaptation of the feature induction technique proposed by Della Pietra et al. (1997). Della Pietra et al. propose a greedy approach for adding induced features to the underlying model. During each iteration, the information gain for each induced feature is computed. The feature with the highest information gain is then added to the model and the entire model is retrained. Although we do not actually induce new features in the present work, we use a similar algorithm for selecting from a large pool of features. Another difference is that the algorithm scores each feature according to any information retrieval metric of interest. The feature that improves the metric the most is the one that is added to the model.

There has also been some information retrieval research into automatically learning ranking function using genetic programming (Fan et al. 2004). These algorithms attempt to find a locally optimal ranking function by iteratively "evolving" a population of ranking functions using mutations and crossovers. Ranking functions are represented as arithmetic trees that consist of arithmetic operators and standard bag-of-words information retrieval features (e.g., term frequency, document length, etc.). The learned ranking functions have been shown to be significantly more effective than baseline ranking algorithms for several data sets (Fan et al. 2004).

Finally, result fusion techniques are another way of combining evidence from multiple types of features (Bartell et al. 1994; Fox and Shaw 1993). If each individual feature is used as a ranking function, then data fusion techniques can be used to determine the best way to combine the rankings. However, using these techniques in this way does not directly address the feature selection problem, which is the primary focus.

2.2 Automatic Feature Selection

As we described before, feature selection techniques are commonly used in the machine learning community. In this section, we describe a feature selection algorithm that can be used with the MRF model. The algorithm is specifically designed to be used for information retrieval tasks.

Feature selection is important for a number of reasons. First, it provides a general, robust way of building models when there is little *a priori* knowledge about the types of features that may be important for a given task or data set. By using a feature selection algorithm, the model designer can focus less on building the best model and can instead focus on designing good features. Second, feature selection can reduce the number of noisy or redundant features in a large feature set. Such features may reduce training efficiency and may result in a model that contains a number of non-identifiable parameters. Non-identifiable parameters are those that cannot be reasonably estimated given the training data. This often results from having redundant or highly correlated parameters. Feature selection helps overcome the problems associated with non-identifiable parameters. Finally, feature selection can provide insights into the important features for a given task or data set. By inspecting the order in which features are selected, we can often learn what characteristics of a given task are the most important or the most exploitable. This knowledge can then be used by the feature engineer to construct better features.

We now describe the automatic feature selection algorithm. While the discussion will focus on how the algorithm can be applied to the MRF model for IR, it should be noted that it can also be applied to a variety of other models. In particular, it can be easily applied to any linear feature-based model.

Let M_t denote the model learned after iteration t. Features are denoted by f and the weight (parameter) associated with feature f is denoted by λ_f. The candidate set of features is denoted by $\mathcal{F}$. The entire set of feature weights for a model is denoted by Λ. A model, then, is represented as set of feature/weight pairs. Finally, we assume that $SCORE(M)$ returns the utility or 'goodness' of model M with respect to some training data. The utility function and the form of the training data largely depends on the underlying task. For example, for *ad hoc* retrieval, it is likely that $SCORE(\cdot)$ would return the mean average precision of using model M against some set of training data, such as TREC topics and relevance judgments. For a homepage finding task, $SCORE(\cdot)$ might be another metric, such as mean reciprocal rank. The important thing to note here is that any utility function, regardless of whether or not it is differentiable with respect to the model parameters, can be used. The ultimate goal of our feature selection algorithm is to select features and set feature weights in such a manner as to maximize the metric imposed by $SCORE(\cdot)$.

The algorithm begins with an empty model (i.e., $M_0 = \{\}$). Then, we temporarily add a feature f to the model. We then hold all weights except λ_f fixed and find the setting for λ_f that maximizes the utility of the augmented model. This step can be done using any number of learning-to-rank techniques or parameter estimation techniques, including the ones described earlier in this chapter. The utility of feature f ($SCORE_f$) is defined to be the maximum utility obtained during training. The feature's utility measures how good the current model would be if the feature were added to it. This process is repeated for every $f \in \mathcal{F}$, resulting in a utility being computed for every feature in the candidate pool. The feature with the maximum utility is then added to the model and removed from $\mathcal{F}$. After the new feature is added, we can, optionally, retrain the entire set of weights. The entire process is then repeated until either some fixed number of features have been added to the

model or until the change in utility between consecutive iterations drops below some threshold. Algorithm 4 provides pseudo-code for this algorithm.

Note that the algorithm is not guaranteed to find the global maximum for $SCORE(M)$. Instead, we are only guaranteed to find a local maxima. Many factors, including properties of $SCORE(M)$, the number of features used, and the properties of the feature used, will affect the quality of the learned model.

Algorithm 4 Feature selection algorithm

$t \leftarrow 0$
$M_t \leftarrow \{\}$
while $SCORE(M_t) - SCORE(M_{t-1}) > \epsilon$ **do**
  for $f \in \mathcal{F}$ **do**
    $\hat{\lambda}_f \leftarrow \arg\max_{\lambda_f} SCORE(M \cup \{(f, \lambda_f)\})$
    $SCORE_f \leftarrow SCORE(M \cup \{(f, \hat{\lambda}_f)\})$
  end for
  $f^* \leftarrow \arg\max_f SCORE_f$
  $M \leftarrow M \cup \{(f^*, \hat{\lambda}_{f^*})\}$
  $\Lambda \leftarrow \arg\max_{\Lambda} SCORE(M)$ (optional)
  $\mathcal{F} \leftarrow \mathcal{F} - \{f^*\}$
  $t \leftarrow t + 1$
end while

2.3 Evaluation

In this section we experimentally evaluate various aspects of the greedy feature selection algorithm.

In order to investigate the strengths and weaknesses of the algorithm, we evaluate its effectiveness on a wide range of *ad hoc* retrieval data sets. The TREC data sets used in the experiments are summarized in Appendix A.

All collections were stopped using a standard list of 418 common terms and stemmed using a Porter stemmer. Only the title portion of the TREC topics are used to construct queries. The primary evaluation metric is mean average precision. Statistical significance is determined using a one-tailed paired *t-test* evaluated at the $p < 0.05$ level.

We now describe the feature candidate pool in terms of the feature representation scheme we described in Sect. 4. For dependence model type, the features may be either *FI* (full independence), *SD* (sequential dependence), or *FD* (full dependence). The clique set type may either be T_{QD}, O_{QD}, or U_{QD}. The weighting functions include LM, BM25, [LM, BM25]-O-[1, 2, 4, 8, 16, or 32], and [LM, BM25]-U-[1, 2, 4, 8, 16, 32, or unlimited]. As we see, the pool is very robust and covers many different types of important features. It includes features that span all three type of

Table 6.1 Training and test set mean average precision values for no retraining and retraining

	No Retrain		Retrain	
	Train	Test	Train	Test
AP	0.1863	0.2266	0.1865	0.2246
WSJ	0.2700	0.3553	0.2703	0.3543
ROBUST04	0.2387	0.3079	0.2391	0.3065
WT10G	0.2344	0.2129	0.2357	0.2140

dependence, use all three types of clique sets, allow both Dirichlet or BM25 weighting, and vary the window sizes for the ordered and unordered window matchings across a wide range of values. In total, after removing trivial and duplicate features, the candidate pool consists of 48 features.

For all of the experiments, features are added until there is no change in training set mean average precision between iterations (i.e., $\epsilon = 0$) or until we have added 5 features. Preliminary experiments showed that adding more than 5 features never resulted in significantly different training or test rest results.

2.3.1 No Retraining vs. Retraining

We wish to analyze what effect, if any, retraining (see Algorithm 4, line 10) has on training and generalization properties of the model. Table 6.1 summarizes the mean average precision obtained on the training and test set when retraining is used and when it is not.

We first investigate whether or not the models learned with retraining vary significantly from those learned without retraining. As Table 6.1 shows, the training set mean average precision values for no retraining and retraining are nearly equivalent for every data set. In fact, the differences are statistically indistinguishable. In addition, we discovered that the same set of features were added regardless of whether or not retraining was done or not. Therefore, it appears as though retraining has little effect on the learned model, both in terms of the features selected and the training set mean average precision.

Next, we study the effect of retraining on the generalization properties of the model. As the test set results in Table 6.1 show, there is very little difference in mean average precision for no retraining versus retraining. The results, again, are statistically indistinguishable for every data set. Hence, retraining does not significantly affect how well the model generalizes to unseen data.

Therefore, given that retraining requires more computational power, has no effect on either the learned model or the generalization properties of the model, we conclude that there is no need to retrain the model each iteration.

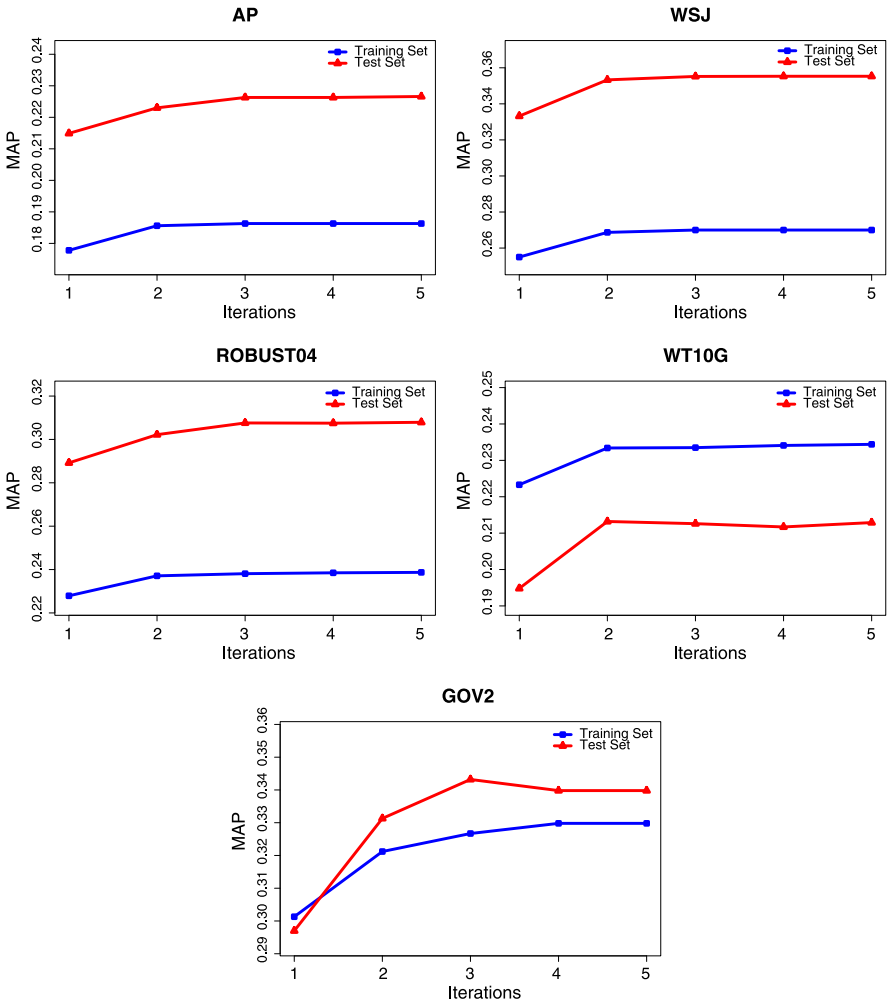

Fig. 6.1 Mean average precision versus number of iterations for the training and test sets of the AP, WSJ, ROBUST04, WT10G and GOV2 data sets

2.3.2 Number of Features

We now analyze how sensitive the models are to the number of parameters, both in terms of potential overfitting, and in terms of test set effectiveness.

Figure 6.1 plots the training and test set mean average precision versus the number of features that have been added to the model. As the figure indicates, there appears to be little, if any overfitting happening. The test set mean average precision never significantly drops as more features are added to the model.

2.3.3 Feature Analysis

The greedy nature of the feature selection algorithm provides us with a mechanism for analyzing the importance of different types of features across data sets. By looking at the order in which features are selected, and the weight assigned to each, we can develop deeper insights into the role that features play for a given task and/or data set.

For example, for the WT10G data set, with no retraining, the features are selected in the following order:

$$(\text{FI}, T_{QD}, \text{BM25}): \quad 0.8138 \tag{6.3}$$

$$(\text{FD}, U_{QD}, \text{LM-U-8}): \quad 0.0001 \tag{6.4}$$

$$(\text{SD}, U_{QD}, \text{BM25-U-unlimited}): \quad 0.0090 \tag{6.5}$$

$$(\text{FD}, U_{QD}, \text{BM25-U-8}): \quad 0.1575 \tag{6.6}$$

$$(\text{SD}, O_{QD}, \text{BM25-O-8}): \quad 0.0196 \tag{6.7}$$

where the numbers after the colons are the weights assigned to each feature in the final model.

As Fig. 6.1 shows, there is a large increase in both training and test set mean average precision after the second feature, LM-U-8, is added to the model. This large increase, which is also exhibited for the GOV2 data set, reiterates the importance of term proximity models for large Web collections (Metzler and Croft 2005). We see that simply adding a single proximity feature increases mean average precision substantially. However, there is a much smaller effect observed after further term proximity/dependence model features are added to the model.

To provide a different example, we consider the order in which features are selected for the WSJ collection. The features selected, in order, are:

$$(\text{FI}, T_{QD}, \text{BM25}): \quad 0.5864 \tag{6.8}$$

$$(\text{SD}, U_{QD}, \text{BM25-U-1}): \quad 0.3746 \tag{6.9}$$

$$(\text{FD}, U_{QD}, \text{BM25-U-32}): \quad 0.0193 \tag{6.10}$$

$$(\text{FI}, T_{QD}, \text{LM}): \quad 0.0196 \tag{6.11}$$

$$(\text{FD}, U_{QD}, \text{BM25-U-unlimited}): \quad 0.0001 \tag{6.12}$$

As with the WT10G model, the first feature selected is the full independence, single term, BM25 feature. In fact, this feature was the first selected for every data set. This is not surprising, however, since the overwhelming importance of single term features has long been understood.

However, no other strong regularities were observed across data sets. This indicates that each data set has unique characteristics that make certain features more discriminative than others. Such characteristics may include things like query

Table 6.2 Comparison of test set mean average precision for language modeling (MRF-FI), BM25, MRF model using language modeling weighting (MRF-SD), MRF model using BM25 weighting (MRF-BM25), and MRF learned using the greedy feature selection algorithm (MRF-FS). A † indicates a statistically significant improvement over *both* the MRF-FI and BM25 models and a ‡ indicates a significant improvement over the MRF-BM25 model

	MRF-FI	BM25	MRF-SD	MRF-BM25	MRF-FS
AP	0.2077	0.2149	0.2128	0.2210	0.2266†‡
WSJ	0.3258	0.3332	0.3429†	0.3512†	0.3553†‡
ROBUST04	0.2920	0.2892	0.3092†	0.3101†	0.3079†
WT10G	0.1861	0.1948	0.2140†	0.2129†	0.2129†
GOV2	0.2984	0.2971	0.3360†	0.3476†	0.3398†

length, noise, document length distribution, and properties of the underlying vocabulary. This suggests that no single model, with a fixed feature set and fixed feature weights, can be applied to every possible task and data set. Instead, adaptive models and techniques, such as the one presented here, can provide a means for automatically and robustly learning the best set of features to use on a task-by-task basis.

2.3.4 Summary of Results

Finally, we compare the retrieval effectiveness of the models automatically learned using the feature selection algorithm (MRF-FS) with several other retrieval models, including language modeling (MRF-FI model), BM25, and two MRF models with hand selected features (MRF-SD and MRF-BM25, as defined in Chap. 3) that we have been shown to be highly effective. For each model, parameters are tuned on the training set to maximize mean average precision. Therefore, every model is properly trained in accordance with the same evaluation metric. This allows us to compare the effectiveness of automatically learned models with models that use manually chosen features and have been proven to be highly effective.

Our results, which are summarized in Table 6.2, support previous observations that show that using MRF models with hand chosen features are generally more effective than bag-of-words models for *ad hoc* retrieval. However, we are interested in how effective the automatically learned models are. For both the AP and WSJ data sets, the mean average precision of the automatically learned model is statistically significantly better than all of the other models, including the MRF models with manually chosen features. The improvement in mean average precision over BM25 for the AP data set is 5.4% and 6.6% on the WSJ data set.

On the ROBUST04, WT10G, and GOV2 data sets, the automatically learned models are statistically significantly better than language modeling and BM25, but statistically indistinguishable from the two models with hand selected features. Despite the lack of a statistically significant improvement over the models with hand selected features, the results still provide evidence that the learned model is highly

effective. Indeed, compared to BM25, the automatically learned model is 6.5% better for ROBUST04, 9.3% better for WT10G, and 14.4% better for GOV2.

Therefore, the results show that the greedy feature selection algorithm produces very effective models that are competitive with, and often significantly better than models with hand selected features. The most important result, however, is that *every* automatically learned model was significantly better than the two state-of-the-art bag-of-words models. This result is very powerful, as it shows that using this framework, models can be automatically learned from a rich set of features and are very likely to be significantly better than the best hand crafted bag-of-words models.

3 Learning to Efficiently Rank

To be successful, search engines must return the most relevant information to the user in a short amount of time. However, efficiency (speed) and effectiveness (relevance) are competing forces that often counteract each other. It is often the case that methods developed for improving relevance incur moderate-to-large computational costs. Therefore, sustained relevance gains must often be counter-balanced by buying more (or faster) hardware, implementing caching strategies if possible, or spending additional effort in low-level optimizations.

It is common for search engines to select a single operating point in the space of all possible efficiency-effectiveness tradeoffs. However, users and information needs are diverse. While most users may want their search results immediately, others may not mind waiting a little extra time if it means their results, on average, would be better. This same idea can be applied to information needs. Certain classes of queries, such as those for simple information needs, are expected to be answered immediately. However, for very complex information needs, users may be willing to incur additional latency for better results. Hence, operating at a "one size fits all" point along the tradeoff curve may not be optimal for all users and queries.

This section explores issues related to the efficiency-effectiveness tradeoff in the context of developing highly effective, highly efficient search engine ranking functions. We describe a framework for automatically learning ranking functions that optimize the tradeoff between efficiency and effectiveness. Traditional learning-to-rank approaches (Liu 2009), have focused entirely on effectiveness. Therefore, it is more appropriate to think of the proposed approach as a learning-to-*efficiently* rank method. We will show that learning to rank (i.e., only optimizing effectiveness) is simply a special case of the proposed framework.

The framework consists of two components. The first is a set of novel metrics for quantifying the tradeoff between efficiency and effectiveness. The second is an approach to optimizing the metrics for a class of linear ranking functions. As we will show, the framework is robust and effective. It can be used to learn a "one size fits all" ranking function, or be used to learn different ranking functions for different classes of users and information needs that may have their own unique efficiency-effectiveness tradeoff requirements.

The remainder of this section, which is based on Wang et al. (2010a), describes the theoretical foundations of the framework and concludes with an empirical evaluation that shows the approach yields models that are both effective and efficient.

3.1 Related Work

There has been a great deal of research devoted to developing efficient and effective retrieval systems. This has given rise to two distinct research threads. The focus of the first thread is on designing effective retrieval models. This has given rise to a steady stream of effectiveness-centric models, such as language models for information retrieval (Ponte and Croft 1998), the BM25 model (Robertson et al. 1994), numerous term proximity models (Metzler and Croft 2005; Büttcher et al. 2006a; Tao 2007; Bai et al. 2008), and learning to rank (Gey 1994; Nallapati 2004; Burges et al. 2005; Liu 2009). The other thread is devoted to building efficient retrieval systems. Improved query execution strategies (Strohman et al. 2005; Anh and Moffat 2006) and advanced index pruning techniques (Carmel et al. 2001; Büttcher et al. 2006b; Ntoulas and Cho 2007) are just two examples of successful research directions along this thread.

The fact that these two threads are almost always investigated exclusively of each other has created a virtual dichotomy in the information retrieval research community. On one side there are researchers who develop highly effective, yet practically infeasible models and methods en masse. On the other side of the dichotomy are the researchers who design blazingly fast, yet spectacularly ineffective systems. One of the goals of the approach is to take a step toward eliminating this dichotomy by taking an efficiency-minded look at building effective retrieval models.

Our problem is quite different from previous work in index pruning (Carmel et al. 2001; Buttcher and Clarke 2005; Büttcher et al. 2006b; Anh and Moffat 2006; Ntoulas and Cho 2007) and query segmentation (Bendersky et al. 2009). The primary goal of index pruning is to create a small index and search over this reduced index to gain better efficiency. In query segmentation, a syntactic parser is used to identify key term dependence features in a query, and only these key features, rather than all term dependence features, are used to retrieve documents. While both techniques are designed for dealing with query latency, these methods do not directly optimize the underlying efficiency and effectiveness metrics, e.g., optimizing index pruning or optimizing segmentation accuracy is not guaranteed to optimize retrieval effectiveness and efficiency, and their tradeoff.

Another way to speed up query evaluation is through caching (Baeza-Yates et al. 2007) (i.e., term posting lists or query search results caching). The problem being addressed here and caching can be viewed as two complementary approaches for improving efficiency in search. Cached postings/results for a given query can be used during the query evaluation stage to improve efficiency, where the ranking function has been specified. In contrast, we learn efficient ranking functions, where a query evaluation strategy and a caching strategy are assumed to be given.

There have been several solutions proposed for dealing with such tradeoffs in various contexts. First, in the machine learning community, it was shown that l_1 regularization is useful for "encouraging" models to have only a few non-zero parameters, thereby greatly decreasing the time necessary to process test instances (Tibshirani 1994). Thus, l_1 regularized loss functions balance between model effectiveness (e.g., mean squared error, classification accuracy, etc.) and efficiency (number of non-zero parameters). However, quantifying efficiency in this way is overly simple and not very flexible. Indeed, the efficiency of most ranking functions cannot be modeled simply as a function of the number of non-zero parameters, since the costs associated with evaluating different features are unlikely to be uniform (e.g., unigram scoring vs. term proximity scoring). The efficiency of a system ultimately depends on the specific implementation, architecture, etc. Therefore, l_1 regularization is too simple to be effective for jointly optimizing the effectiveness and efficiency of ranking functions.

In a similar direction, Collins-Thompson and Callan (2005) investigated strategies for robust query expansion by modeling expansion term selection and weighting using convex programming. Their model included a variant of l_1 regularization that imposed a penalty for including common terms in the expanded query, since such terms would likely increase query execution time. This was the first effort that we are aware of that modeled efficiency in a search engine-specific manner. However, we note that query expansion and constructing ranking functions are two different problems and hence present different challenges. Furthermore, in this work, we model system efficiency using actual query execution times, instead of simple surrogates, such as term frequency, that may or may not accurately model the actual efficiency of the underlying retrieval system.

Finally, the work can be viewed as an enhancement of existing learning-to-rank strategies, which have, until now, focused exclusively on effectiveness. We expect the exploitation of efficiency in constructing effective ranking functions will allow for rapid development of highly effective *and* efficient retrieval models.

3.2 Tradeoff Metrics

In this section we describe a class of *tradeoff metrics* that consider both effectiveness and efficiency. We begin by defining a general class of efficiency functions and describing how different functions yield different tradeoffs.

3.2.1 Measuring Efficiency

There are many different ways to measure the efficiency of a search engine, such as query execution time and queries executed per second. We are primarily interested in measuring how efficient a ranking function is at producing a ranked list for a *single query*. Throughout the remainder of this section, we will assume that the

measure of interest is query execution time, although any other query-level measure of efficiency could also be used.

Query execution times are, in theory, unbounded. This makes them difficult to work with from an optimization point of view. Instead, we would like to map query execution times into the range [0, 1]. We accomplish this by defining a function $\sigma(\cdot) : \mathbb{R}^+ \rightarrow [0, 1]$ that takes a query execution time, denoted by $\tau(Q)$ as input and returns an efficiency metric in the range [0, 1], where 0 represents an inefficient ranking function and 1 represents an efficient ranking function.

We now define four different efficiency metrics. Each metric differs by how $\sigma(\cdot)$ is defined.

Constant The most trivial efficiency metric is defined as $\sigma(Q) = c$, for $c \in [0, 1]$. This constant efficiency metric is always the same, regardless of the query execution time. This is the default assumption made by previous learning-to-rank approaches, which ignore efficiency altogether.

Exponential Decay This loss function is defined as:

$$\sigma(Q) = \exp\big(\alpha \cdot \tau(Q)\big), \tag{6.13}$$

where $\alpha < 0$ is a parameter that controls how rapidly the efficiency metric decreases as a function of query execution time. If a large (negative) decay rate (i.e., α) is specified, then the metric will drop off very quickly, penalizing all but the fastest query execution times.

Step Function Often it is necessary to incorporate query execution time preferences into the efficiency metric. For instance, users may have a certain tolerance level for query execution time, such that they would expect the time to be less than a target t milliseconds for each query. A step function can naturally account for this requirement, as follows:

$$\sigma(Q) = \begin{cases} 1, & \text{if } \tau(Q) \leq t, \\ 0, & \text{if } \tau(Q) > t. \end{cases} \tag{6.14}$$

The step function metric is maximal (1) when query execution time is less than t and minimal (0) otherwise.

Step + Exponential Decay If query execution time exceeds the threshold t in the step function efficiency metric, but only by a small amount, the metric assigned will still be 0, which may be overly harsh. Instead, it may be more reasonable to define a soft loss-like function, as follows:

$$\sigma(Q) = \begin{cases} 1, & \text{if } \tau(Q) \leq t, \\ \exp(\alpha \cdot (\tau(Q) - t)), & \text{if } \tau(Q) > t, \end{cases} \tag{6.15}$$

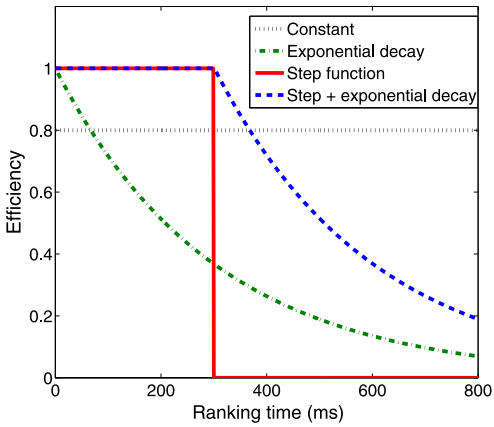

Fig. 6.2 Efficiency functions. Constant and exponential decay are self explanatory. The step function and step + exponential function can model time preferences (threshold $t = 300$ ms here); when exceeding the time requirement, the ranking model either gets zero efficiency value or an exponentially lower efficiency value (respectively)

where $\alpha < 0$. The resulting function is a step function up until the threshold t and an exponential decay after time t with parameter α.

Figure 6.2 summarizes the four efficiency metrics just described. Note that there are many other ways to define $\sigma(\cdot)$ beyond those explored here. The best functional form for a given task will depend on many factors, including data set size, hardware configuration, among others.

3.2.2 Measuring Effectiveness

There has been a great deal of research into evaluating the effectiveness of information retrieval systems. Therefore, we simply make use of existing effectiveness measures here. We define the effectiveness of a query Q as $\gamma(Q)$.

As with the efficiency metrics, we are primarily interested in effectiveness measures with range $[0, 1]$. Most of the commonly used effectiveness metrics satisfy this property, including precision, recall, average precision, and NDCG. We will exclusively focus on average precision as the effectiveness metric of interest, although any of the above metrics can be substituted in the framework without loss of generality.

3.2.3 Efficiency-Effectiveness Tradeoff Metric

Our goal is to automatically learn ranking models that achieve an optimal middle ground between effectiveness and efficiency. However, before we can learn such a well-balanced model, we must define a new metric that captures the tradeoff. The metric, which we call Efficiency-Effectiveness Tradeoff (EET), is defined for a query Q as the weighted harmonic mean of efficiency $\sigma(Q)$ and effectiveness $\gamma(Q)$:

$$\text{EET}(Q) = \frac{(1+\beta^2)\cdot(\gamma(Q)\cdot\sigma(Q))}{\beta^2\cdot\sigma(Q)+\gamma(Q)}, \tag{6.16}$$

where β is a parameter that controls the relative importance between effectiveness and efficiency. In this work, we set $\beta = 1$, which weighs both equally, but other settings can be trivially applied in the approach as well.

Given a ranking model R, the value of EET is computed for each query. To quantify the average tradeoff performance across N queries for a given ranking function, we define the following metric:

$$\text{MEET}(R) = \frac{1}{N} \sum \text{EET}(Q) \tag{6.17}$$

which is simply the mean EET value for the set of N queries.

It should now be clear that different choices of efficiency metrics will have a direct influence on MEET. For instance, a sharply decaying exponential efficiency metric represents a low tolerance for inefficient ranking models. Under such a function, the efficiency metric for a ranking function with high query execution time will likely be extremely low, resulting in a small MEET value, even if the ranking function is effective. On the other hand, if the efficiency function decays slowly or is constant, a ranking function with high effectiveness will also likely have a large MEET value.

Different combinations of efficiency metric and effectiveness metric will give rise to different MEET instantiations. Therefore, MEET is not a single metric, but a family of tradeoff metrics that depends on an efficiency component $\sigma(Q)$, an effectiveness component $\gamma(Q)$, and a tradeoff factor β.

3.3 Model

We focus our attention on linear feature-based ranking functions, such as those are derived within the MRF retrieval framework, which have the following form:

$$S(Q, D) = \sum_j \lambda_j f_j(Q, D), \tag{6.18}$$

where Q is a query, D is a document, $f_j(Q, D)$ is a feature function, and λ_j is the weight assigned to feature j.

Since we are also interested in optimizing for efficiency, we would like a mechanism for altering the efficiency characteristics of the ranking function. The most straightforward way to accomplish this is to eliminate one or more features from the ranking function. A logical way of choosing features to eliminate are those with small weights, as is done with l_1 regularization. We adopt a slight variant of this approach, where we assume that each weight λ also takes on a parametric form, as follows:

$$\lambda_j(Q) = \sum_i w_i g_i(Q), \tag{6.19}$$

where $g_i(Q)$ is a meta-feature that takes Q as input (discussed a bit later), and w_i is the weight assigned to the meta-feature. Notice that the weights λ are now *query dependent*, which means they can adapt better to different query scenarios via the feature functions g_i. We will show shortly that allowing λ to depend on Q provides an intuitive way to prune features. This is precisely the same query-dependent feature weighting model that was described in Chap. 5.

Plugging these query-dependent weights into the original linear ranking function (Eq. 6.18) gives us the following ranking function:

$$S(Q, D) = \sum_i w_i \sum_j g_i(Q) f_j(Q, D) \tag{6.20}$$

which is still a linear ranking function, but now with respect to w_i, which are the global, query-independent free parameters that must be learned.

As a concrete example of a highly effective ranking function that takes on this functional form, we consider the *weighted sequential dependence* (WSD) ranking function that was described in Chap. 5. The WSD ranking function is formulated as:

$$\begin{aligned} S(Q, D) = & \sum_i w_i^t \sum_{q \in Q} g_i^t(q) f_T(q, D) \\ & + \sum_i w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_O(q_j, q_{j+1}, D) \\ & + \sum_i w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_U(q_j, q_{j+1}, D), \end{aligned} \tag{6.21}$$

where the features f_T and g_i^t are defined over query unigrams, and f_O, f_U, and g_i^b are defined over query bigrams. Table 5.3 shows how the features f_T, f_O, and f_U are defined, while Table 5.2 summarizes the meta-features g_i^t and g_i^b. Hence, this specific ranking function consists of three general components: (1) a unigram term score, (2) a bigram phrase score, and (3) a bigram proximity score.

While the effectiveness of the ranking function depends on the weights w, models of this form provide a natural mechanism for eliminating features in a query-dependent manner, thereby improving efficiency. We drop features according to the magnitude of $\lambda_i(Q)$, the query-dependent weight assigned to feature i. If $\lambda_i(Q)$ is nearly zero, then feature i is unlikely to have a significant impact on the final ranking. Therefore, it should be safe to drop feature i from the ranking function, thereby increasing the efficiency of the model, with minimal impact on effectiveness. This suggests the general strategy of pruning features if $|\lambda_i(Q)| \leq \epsilon$, where ϵ is a pruning threshold.

However, in this work, we are dealing with a model that we have specific domain knowledge about, and therefore use a more suitable pruning strategy. Previous work by Lease (2009) demonstrated that unigrams have more positive impact on retrieval effectiveness than bigrams; hence, we only prune bigram features from the WSD

ranking function. Bigram features are pruned if they satisfy the condition

$$\frac{\lambda(q_i, q_{i+1})}{\lambda(q_i) + \lambda(q_{i+1})} \leq \epsilon. \tag{6.22}$$

This condition says that if the ratio between the bigram feature weight and the sum of individual unigram feature weights is less than ϵ, then the bigram is eliminated. Preliminary experiments found the general strategy of pruning according to $|\lambda_i(Q)| \leq \epsilon$ to be effective, but found this ranking function-specific strategy to yield superior results. Therefore, it is likely that different ranking functions will require different pruning strategies to be maximally effective.

3.4 Parameter Estimation

We now describe the method for automatically learning the parameters of the model from training data. We must not only learn the parameters w, but also the concept pruning threshold ϵ. Although there are many learning-to-rank approaches for learning a linear ranking function (i.e., estimating w), the optimization problem is complicated by the fact that we also have to learn the best ϵ, which is directly tied to the efficiency of the ranking function. Since the relationship between the metric and ϵ cannot be modeled analytically, we are forced to directly estimate the parameters using a non-analytical optimization procedure.

We used a simple optimization procedure that directly optimizes MEET: the coordinate-level ascent algorithm that was described earlier in this chapter. The algorithm performs coordinate ascent in the MEET metric space. Each (single dimensional) optimization problem is solved using a simple line search. Given that the MEET space is unlikely to be convex, there is no guarantee that this greedy hill-climbing approach will find a global optimum, but, as we will show, it tends to reliably find good solutions for this particular optimization problem. The final solution to the optimization problem is a setting of the parameters w and a pruning threshold ϵ that is a local maximum for the MEET metric.

3.5 Experimental Results

To illustrate the benefits of the approach across a diverse set of document collections, we used three TREC Web collections (WT10g, GOV2, and ClueWeb09). We compare the model, which we call the efficient sequential dependence model (ESD), to two baseline models. One is the bag-of-words query-likelihood model (Ponte and Croft 1998) (QL), with Dirichlet smoothing parameter $\mu = 1000$. The other is the less efficient, but more effective sequential dependence variant of the MRF model (SD). The SD model is a special case of the WSD and ESD models. The best practice implementation of the SD model uses the same features and functional form as

Table 6.3 Comparison between models under step + exponential efficiency function (slow decay on top, fast decay on bottom); parameters t (time threshold) and α (decay rate) are shown in the column headings for each collection. Symbol * denotes significant difference with QL; † denotes significant difference with SD. Percentage improvement shown in parentheses: over QL for SD, and over QL/SD for ESD

"Slow" Decay Rate									
	WT10g			GOV2			ClueWeb09		
	Time	MAP	MEET	Time	MAP	MEET	Time	MAP	MEET
QL	0.168	0.2151	0.2175	1.97	0.3194	0.3196	4.09	0.2075	0.2055
SD	0.401	22.43*	0.2212	7.09	0.3357*	0.3291	13.46	0.2168	0.2141
ESD	0.235	$0.2404^{*}_{\dagger}$	$0.2303^{*}_{\dagger}$	6.42	$0.3474^{*}_{\dagger}$	$0.3394^{*}_{\dagger}$	11.18	0.2234*	0.2214

"Fast" Decay Rate									
	WT10g			GOV2			ClueWeb09		
	Time	MAP	MEET	Time	MAP	MEET	Time	MAP	MEET
QL	0.168	0.2151	0.2103	1.97	0.3194	0.3187	4.09	0.2075	0.2053
SD	0.401	0.2243*	0.1963*	7.09	0.3357*	0.3177	13.46	0.2168	0.2126
ESD	0.215	0.2342*	$0.2155^{*}_{\dagger}$	5.46	0.3365*	0.3258	8.55	0.2124	0.2108

the WSD model, but sets $w_5^t = 0.82$, $w_5^b = 0.09$. All of the other parameters are set to 0, yielding *query independent* λ_i weights. The SD model does not prune features, meaning that all features are evaluated for every query.

Effectiveness is measured in terms of mean average precision (MAP), although as previously noted a variety of other effectiveness metrics can be substituted. As for efficiency, we explored several different efficiency functions, and analyzed the resulting impacts on the tradeoff between efficiency and effectiveness (detailed below). When training the model, we directly optimized the MEET metric. A Wilcoxon signed rank test with $p < 0.05$ was used to determine the statistical significance of differences in the metrics.

In the remainder of this section we describe the performance of the model in terms of its ability to optimally balance effectiveness and efficiency. We show the impact of different efficiency functions on the learned models and also present an analysis of the distribution of query times, demonstrating reduced variance. Finally, we show that with specific efficiency functions, the learned models converge to either baseline query likelihood or the weighted sequential dependence model, thus illustrating the generality of the framework in subsuming ranking approaches that only take into account effectiveness.

3.5.1 Tradeoff Between Effectiveness and Efficiency

Table 6.3 presents results of two sets of experiments using the step + exponential function, with what we subjectively characterize as "slow" decay and "fast" de-

cay. The time threshold t (below which efficiency is one) was chosen to be roughly halfway between the QL and SD running times for each collection. Due to the differences in collection size, it is unlikely that a common decay rate (α) is appropriate for all collections. Therefore, we manually selected a separate decay rate for each collection. Both t and α are shown in the column headings of Table 6.3. The fast decay function penalizes low efficiency ranking functions more heavily, thus a highly efficient ranking function with reasonable effectiveness is preferred over a less efficient function with potentially better effectiveness. With the slow decay function, effectiveness plays a greater role.

For both fast and slow decay, we compared the model described here (ESD) with the query-likelihood model (QL) and the sequential dependence model (SD) in terms of query evaluation time, MAP, and MEET. In both tables, percentage improvements for MAP and MEET are shown in parentheses: over QL for SD, and over QL/SD for ESD. Statistical significance is denoted by special symbols.

As expected, the mean query evaluation time for ESD is greater than that of QL, but less than that of SD for both sets of experiments. Furthermore, the mean query evaluation time for ESD is lower for the fast decay rate than for the slow decay rate, which suggests that the efficiency loss function is behaving as expected. In the learned models, ESD is 41.4%, 9.4%, and 16.9% faster than SD for the slow decay rate on WT10g, GOV2, and ClueWeb09, respectively; ESD is 46.4%, 23.0%, and 36.5% faster than SD for the fast decay rate on the same three collections, respectively. Once again, this makes sense, since the fast decay rate penalizes inefficient ranking functions more heavily.

In terms of mean average precision, in five out of the six conditions, the ESD model was significantly better than baseline QL. In the one condition in which this was not the case (ClueWeb09 with fast decay), SD was not significantly better than QL either. While the ESD model is much more efficient than the SD model, it is able to retain the *same* effectiveness as SD, and in some cases actually performs *significantly better* (in the case of WT10g and GOV2 for slow decay rate). We believe that this result demonstrates the ability of the framework to select a more optimal operating point in the space of effectiveness-efficiency tradeoffs than previous approaches.

In terms of MEET, the ESD model outperforms QL and SD, although gains are not consistently significant. Interestingly, we note that SD has a lower MEET score than default QL in two out of three collections when the fast decay rate is used. This suggests that the default formulation of the sequential dependence model trades off a bit too much efficiency for effectiveness, at least based on the metrics.

Lastly, note that setting the time target t in the efficiency function implies that the ranking model will be penalized by an exponential decay in efficiency if its query ranking time exceeds t. This is a soft penalization factor, which contrasts with the more harsh step function where efficiency is assigned a zero value for time exceeding t. An implication of using this soft efficiency loss function is that for a ranking function with time $> t$, if it is *highly* effective for user queries, it may still have a reasonable tradeoff value, because essentially, its high effectiveness compensates for the loss in efficiency. This fact is also confirmed by results shown

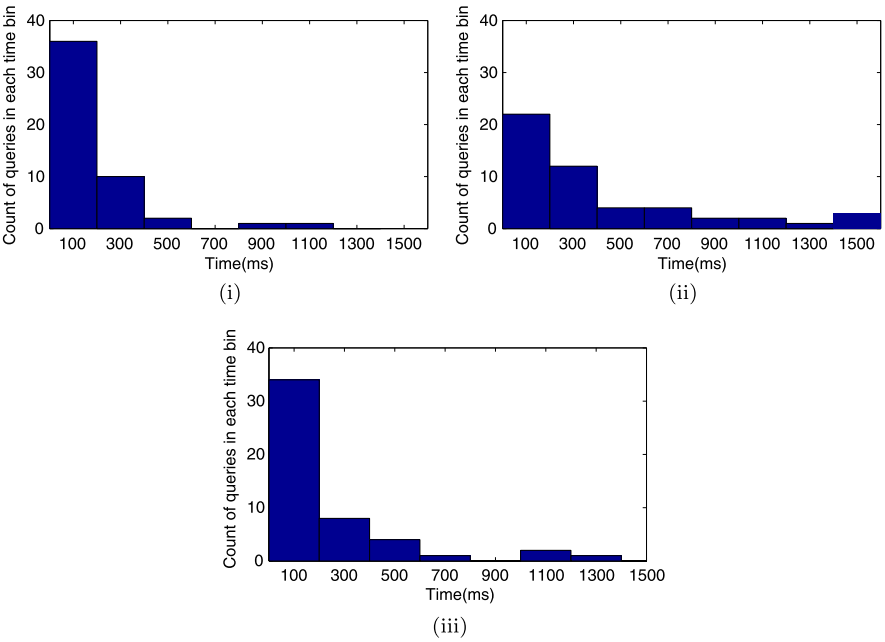

Fig. 6.3 Distribution of query execution times for WT10g queries for (**i**) query likelihood (QL); (**ii**) sequential dependence model (SD); (**iii**) efficient sequential dependence model (ESD)

in Table 6.3, where the average query time of ESD is consistently greater than the time threshold t.

3.5.2 Analysis of Query Latency Distribution

Another benefit of the framework is that learned models exhibit low variance in query execution times. Figure 6.3 plots histograms of query execution for QL, SD, and ESD on the WT10g collection. The ESD model was trained using the step + exponential (fast decay) efficiency function. Distributions for the other conditions look similar, and therefore we omit in the interest of space.

We can see that most queries with baseline QL are evaluated in a short amount of time, with a small number of outliers. The sequential dependence model has a heavier tail distribution with increased variance in query execution times: most queries still finish relatively quickly, but a significant minority of the queries take much longer to evaluate. The ESD model reduces the number of long-running queries so that the histogram is less tail heavy, which greatly improves the observed variance of query execution times. This improved behavior is due to the fact that the model considers efficiency as well as effectiveness, hence penalizes long-running queries, even if they are more effective. Note that although query likelihood has the most desirable query execution profile, it comes at the cost of effectiveness. Experiments in

Table 6.4 Comparison of SD and ESD under constant efficiency (i.e., only effectiveness is accounted for in the tradeoff metric)

	WT10g			GOV2			ClueWeb09		
	Time	MAP	MEET	Time	MAP	MEET	Time	MAP	MEET
SD	0.401	0.2243	0.2244	7.09	0.3357	0.3327	13.46	0.2168	0.2142
ESD	0.425	$0.2411_{\dagger}$	$0.2334_{\dagger}$	7.13	$0.3435_{\dagger}$	$0.3408_{\dagger}$	13.87	0.2243	0.2226

the previous section showed that ESD is at least as effective as SD, but much more efficient. The distribution of query execution times further supports this conclusion.

Why is reduced variance in query execution time important? For real-world search engines, it is important to ensure that a user, *on average*, gets good results quickly. However, it is equally important to ensure that *no user* waits too long, since these represent potentially dissatisfied users who never come back. A basic principle in human–computer interactions is that the user should never be surprised, and that system behavior falls in line with user expectations. Reducing the variance of query execution times helps us accomplish this goal.

Furthermore, from a systems engineering point of view, lower variance in query execution time improves load balancing across multiple servers. In real-world systems, high query throughput is achieved by replicated services across which load is distributed. If variance of query execution times is high, simple approaches (e.g., round-robin) can result in uneven loads (consider, for example, that in SD one query can take an order of magnitude longer than another to execute). Therefore, the reduced variance exhibited by the learned models is a desirable property.

3.5.3 Relationships to Other Retrieval Models

Finally, we demonstrate that previous ranking models that consider effectiveness only can be viewed as a special case of the family of ranking functions that account for both effectiveness and efficiency. More specifically, the flexible choices for efficiency functions used in the general framework described here can capture a wide range of tradeoff scenarios for different effectiveness/efficiency requirements. For instance, if we care more about efficiency than effectiveness, then we can set the time threshold in the efficiency function to be low, which forces us to learn a ranking function with high efficiency. On the other hand, if the focus is to learn the most effective ranking function possible (disregarding efficiency), then we can use a constant efficiency value. We would expect that in the first case, the learned model would look very similar to baseline query likelihood (efficient but not effective). Correspondingly, we would expect that in the latter case, the learned model would look very similar to the sequential dependence model (effective but not efficient). For particular choices of the efficiency function, the learned models should converge to (i.e., acquire similar parameter settings as) existing models that encode a specific effectiveness/efficiency tradeoff.

Table 6.5 Comparison of QL and ESD under step efficiency functions. A step function with $t = 3$ s is used for ClueWeb09 and GOV2, and a step function with $t = 100$ ms is used for WT10g

	WT10g			GOV2			ClueWeb09		
	Time	MAP	MEET	Time	MAP	MEET	Time	MAP	MEET
QL	0.168	0.2151	0.1337	1.97	0.3194	0.2582	4.08	0.2075	0.1602
ESD	0.145	0.2150	0.1350	1.93	0.3163	0.2539	3.68	0.2070	0.1602

Table 6.4 compares ESD to the SD model with a constant efficiency function. The model described here, when trained with constant efficiency values, is equivalent to the WSD model (Bendersky et al. 2010). The ESD model in this case significantly outperforms SD in MAP and MEET scores; the differences are significant in two of the three collections.

Similarly, Table 6.5 illustrates the relationship of ESD to QL under a step efficiency function with low time targets ($t = 100$ ms is used for WT10g and $t = 3$ s is used for ClueWeb09 and GOV2). Step efficiency functions heavily penalize long query execution times, so the model essentially converges to simple bag of words. An interesting observation is that while retaining similar effectiveness as the QL model, the ESD model achieves a better time efficiency than QL due to its joint optimization of effectiveness and efficiency (allowing the model to prune query terms that have little impact on effectiveness, but nevertheless have an efficiency cost).

Appendix A
Data Sets

In order to properly analyze the various dimensions of retrieval effectiveness, it is important to evaluate new models and techniques against a diverse set of data sets. We make use of TREC collections in all of the experiments. Therefore, we begin this Appendix by describing the anatomy of a TREC data set. We then provide the specific details of the data sets used in the experiments throughout this work.

1 Anatomy of a TREC Data Set

Every TREC data set consists of a set of *documents*, *topics*, and *relevance judgments*. We now provide an overview of each.

TREC documents, which are in SGML format, contain a number of metadata fields, as well as content. See Fig. A.1 for an example TREC document. Examples of metadata fields include `DOCNO`, which is a unique document identifier and `HEAD`, which is the headline of the news article. The document's textual content is contained within the `TEXT` field. In the experiments, we throw away all of the metadata, except for the document identifier, and index the content contained in the `TEXT` field[1].

A TREC topic typically consists of a `title`, `description`, and a `narrative`, although some topics may contain other fields. See Fig. A.2 for an example TREC topic. It is important to note that a topic represents an information need and is not a query itself. Instead, researchers must distill a query from a topic. There are a number of ways of doing this, but the most common way is to use one of the fields as the query. In most of the experiments done throughout this work, we distill a query by using only the text that appears in the title field.

Finally, relevance judgments are provided. For each topic, a set of documents are manually judged for relevance. Different definitions and scales of relevance are used

[1] TREC web documents do not have a `TEXT` field. For these document, we index all of the text, except that contained in the `DOCHDR` field.

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8,

<DOC>
<DOCNO> AP890101-0010 </DOCNO>
<FILEID>AP-NR-01-01-89 1123EST</FILEID>
<FIRST>r i AM-Thatcher-Women 01-01 0287</FIRST>
<SECOND>AM-Thatcher-Women,0295</SECOND>
<HEAD>Thatcher Says Male Prime Ministers May Eventually Be Fashionable Again</HEAD>
<DATELINE>LONDON (AP) </DATELINE>
<TEXT>
Prime Minister Margaret Thatcher, who made history
in 1979 when she became Europe's first woman prime minister, noted
Sunday she is not alone and joked that male politicians may one day
come back into fashion.
"We're getting more women prime ministers," she said in a
television interview, referring to the recent election of Benazir
Bhutto as prime minister of Pakistan.
"And don't forget . . . Mrs. Gandhi was a very able, charming,
formidable prime minister of India."
Mrs. Thatcher now is the longest serving leader in the West.
Before she came to power, women had governed in Sri Lanka and
Israel. Part of the British leader's tenure in office coincided with
that of Mrs. Gandhi, who was assassinated in 1984.
"I think male prime ministers one day will come back into
fashion," she joked with interviewer David Frost on Britain's
commercial TV-am channel.
Asked about combining her job and her domestic life with her
husband, Denis, a retired oil executive, she said "women have run
both the home and work for a very long time."
"I mean, every working wife knows that if you decide to have
steak and kidney pie for supper, it's no better if you took 20
minutes thinking about it than if you took 20 seconds."
Frost recalled Mrs. Thatcher's comment "I never did understand
men," during an acrimonious meeting last year with fellow leaders
of the European Economic Community. Asked if she understood men
better now, Mrs. Thatcher replied:
"It may not be understanding of the deepest kind, but I do know
what they're likely to do and say. So, one has a certain
predictability about it."
</TEXT>
</DOC>

Fig. A.1 Example TREC document

```
<top>
<num> Number: 763

<title> Hunting deaths

<desc> Description:
Give information on human deaths associated with hunting for game.

<narr> Narrative:
Accidental deaths, murders, and suicides are relevant. Deaths can be
from any cause. Fatalities of people not in the hunting party are
relevant, but the deaths must be connected with hunting. Relevant
hunting must be for live prey. Deaths related to submarine hunting
are not relevant.

</top>
```

Fig. A.2 Example TREC topic

Fig. A.3 Portion of a TREC relevance judgment file. The format of each line is `query-id 0 doc-id judgment`. Judgments of 0, 1, and 2 refer to non-relevant, relevant, and highly relevant, respectively

```
763 0 GX018-79-11508357 2
763 0 GX018-79-15198972 0
763 0 GX019-18-8997173 0
763 0 GX019-40-2241667 0
763 0 GX019-56-11902532 1
763 0 GX019-97-10746106 0
763 0 GX020-45-4344240 0
763 0 GX020-99-3364231 0
763 0 GX021-19-6737921 0
763 0 GX021-41-4253461 0
```

for different tasks. Please refer to Sects. 6 and 7 for examples. Furthermore, due to limited resources, not all documents are judged for every topic. Instead, TREC uses a number of techniques, including document pooling, in order to reduce the number of documents judged, while ensuring the test collection is reusable (Harman 1993; Spärck Jones and van Rijsbergen 1976). Figure A.3 shows an excerpt from a TREC relevance judgment file. These judgments are used as the "ground truth" for the purposes of evaluation, which we cover in Appendix B.

2 Summary of Data Sets

Now that we have explained the composition of a TREC data set, we summarize the details of the data sets considered in this work. First, Table A.1 gives an overview

Table A.1 Overview of TREC collections and topics used in most of the experiments

Name	Description	# Docs	Train Topics	Test Topics
WSJ	Wall St. Journal '87–'92	173,252	51–150	151–200
AP	Associated Press '88–'90	242,918	51–150	151–200
ROBUST04	Newswire articles	528,155	301–450	601–700
WT10g	Small web crawl	1,692,096	451–500	501–550
GOV2	2004 crawl of .gov domain	25,205,179	701–750	751–800

Table A.2 TREC data sets used throughout the book. The disk numbers refer to the TREC volumes used to construct the index

	Disks 1, 2	Disk 3	Disks 4, 5
Num. Docs	741,856	336,310	556,077
Training topics	101–150	51–100	301–350
Test topics	151–200	101–150	401–450

of five primary data sets used in the experiments. The table includes the data set name, a short description, the number of documents in the data set, the topics used for training and the topics used for testing. As we see, these data sets vary in size, homogeneity, and noisiness. The AP and WSJ data sets are small collections of newswire articles from a single source. Therefore, they are homogeneous and contain very little noise. The ROBUST04 data set is medium sized and consists of news stories from a number of sources, thus making it less homogeneous, and not very noisy. Finally, the WT10g and GOV2 data sets are (very) large web crawls, thus making them both heterogeneous and noisy. Hence, the data sets are diverse across a number of characteristics, making this a suitable evaluation testbed. Nearly all of the experiments done throughout this work use the data sets listed in this table.

Another set of data sets is also considered, but only used for one experiment. The data sets shown in Table A.2 are used in the experiments throughout the book. All of these data sets consist primarily of news articles, are somewhat heterogeneous, and contain very little noise. The collections are also relatively small, as well. The table also lists the TREC topics used for training and testing purposes.

Appendix B
Evaluation Metrics

This appendix provides a brief summary of metrics that are commonly used to evaluate the effectiveness of information retrieval systems. Each of these evaluation measures are rank-based. That is, they depend exclusively on the ranking of documents produced by a system. The scores of the documents do not influence the metrics in any way. In addition, a binary model of relevance is assumed, which states that an item is either *relevant* to a request or it is *not relevant*.

Given a ranked list of documents in response to a query, we define the function R such that $R(i) = 1$ if the document at rank i is relevant and $R(i) = 0$ if the document at rank i is not relevant. Furthermore, as a matter of convenience, we define the function $R(1, k)$ that counts the number of relevant documents between rank 1 and k (inclusive) as

$$R(1, k) = \sum_{i=1}^{k} R(i). \tag{B.1}$$

In Table B.1 the functional forms for precision at rank k, R-Prec, success at rank k, average precision, and reciprocal rank are given.

Precision at rank k measures the proportion of relevant documents that are ranked in the top k, whereas success at rank k is 1 if *any* relevant documents appear in the top k and 0 otherwise. Typical values of k range from 5 to 100. Precision at rank 5 or 10, for example, is typically used to evaluate a system's ability to return relevant documents at the top of the ranked list, which is important for many applications, including web search.

R-prec is a special case of precision at rank k. R-Prec computes the precision at rank $|R|$, where $|R|$ is the total number of judged relevant documents. This measure is more adaptive, as it computes precision to a different depth for every query.

Average precision can be thought of as a weighted precision measure that gives higher weight to relevant documents that appear near the top of the ranked list. The measure is computed by averaging the precision at k for every k such that the document at rank k is relevant. Average precision is standardly computed to a depth

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8,

Table B.1 Summary of common information retrieval evaluation metrics, where $R(1,k)$ is defined in Eq. B.1 and $|R|$ is the total number of judged relevant documents

Metric Name	Functional Form
Precision at rank k	$\text{P@}k = \frac{R(1,k)}{k}$
Success at rank k	$\text{S@}k = \delta(R(1,k) \geq 1)$
R-Prec	$\text{R-Prec} = \frac{R(1,\|R\|)}{\|R\|}$
Average Precision	$\text{AvgP} = \frac{1}{\|R\|} \sum_{i:R(i)=1} \frac{R(1,i)}{i}$
Reciprocal Rank	$\text{RR} = \max\{\frac{1}{i} : R(i) = 1\}$

of 1000 documents[1]. Any relevant documents that do no appear in the ranked list are assumed to be at rank "infinity", and therefore contribute nothing to the average. Therefore, the measure rewards systems that rank relevant documents high in the ranked list and those that return more relevant documents. In this way, the measure implicitly accounts for both precision and recall (coverage of relevant documents). Average precision is typically used to evaluate *ad hoc* retrieval tasks and other tasks where both precision and recall are important factors.

The last measure, reciprocal rank, is computed according to the reciprocal of the highest ranked relevant document. Note that the measure quickly decays as the rank of the first relevant document increases (e.g., 1, 0.5, 0.33, 0.25, 0.2, etc.). We note that reciprocal rank is a special case of average precision when there is only a single relevant document for a given request. This measure is commonly used to evaluate known-item finding queries, where it is critical to return the relevant web page very high in the ranked list.

Each of these measures we just described are computed on a query-by-query basis. However, for the sake of comparison, we often need to compute a single measure from a set of query-by-query measures. Given a set of queries, the *arithmetic average* or the *geometric average* are often used to combine, or aggregate, the measures

Table B.2 Overview of aggregate measures. For each aggregate measure we show how it is computed. Here, N refers to the total number of queries being aggregated and q_i is the ith query in the set. Notice that GMAP is zero if any query has an average precision of zero. In order to correct this, $\text{AvgP}'(q_i)$ is defined to be $\max(\text{AvgP}(q_i), 00001)$

Name	Value
Precision at rank k	$\text{P@}k = \frac{1}{N} \sum_{i=1}^{N} \text{P@}k(q_i)$
Success at rank k	$\text{S@}k = \frac{1}{N} \sum_{i=1}^{N} \text{S@}k(q_i)$
R-Prec	$\text{R-Prec} = \frac{1}{N} \sum_{i=1}^{N} \text{R-Prec}(q_i)$
Mean Average Precision	$\text{MAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AvgP}(q_i)$
Geometric Mean Average Precision	$\text{GMAP} = \prod_{i=1}^{N} \text{AvgP}'(q_i)^{1/N}$
Mean Reciprocal Rank	$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \text{RR}(q_i)$

[1] Throughout this work, all evaluation metrics are computed to a depth of 1000.

computed on a query-by-query basis. Table B.2 summarizes the most common aggregate measures used in information retrieval. In order to avoid confusion, the most commonly used names of the evaluation metrics were used. This results in aggregate measures having the same name as their non-aggregate counterparts (e.g., R-Prec is the name of both the aggregate and non-aggregate measure). We note, however, that throughout this work, it should be clear from the context of the discussion whether the measure in question has been computed for a single query or if it was aggregated over a set of queries.

References

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 19–26).

Amati, G., & van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, *20*(4), 357–389.

Anh, V. N., & Moffat, A. (2005). Simplified similarity scoring using term ranks. In *Proc. 28th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 226–233).

Anh, V. N., & Moffat, A. (2006). Pruned query evaluation using pre-computed impacts. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 372–379).

Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., & Silvestri, F. (2007). The impact of caching on search engines. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '07* (pp. 183–190). New York: ACM.

Bai, J., Chang, Y., Cui, H., Zheng, Z., Sun, G., & Li, X. (2008). Investigation of partial query proximity in web search. In *Proc. 17th international conference on World Wide Web* (pp. 1183–1184).

Baron, J. R., Lewis, D. D., & Oard, D. (2006). TREC 2006 legal track overview. In *Proc. 15th intl. conf. on World Wide Web*

Bartell, B. T., Cottrell, G. W., & Belew, R. K. (1994). Automatic combination of multiple ranked retrieval systems. In *Proc. 17th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 173–181).

Bendersky, M., & Croft, W. B. (2008). Discovering key concepts in verbose queries. In *Proc. 31st ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

Bendersky, M., Croft, W. B., & Smith, D. A. (2009). Two-stage query segmentation for information retrieval. In *Proc. 32nd ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

Bendersky, M., Metzler, D., & Croft, W. B. (2010). Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM international conference on Web search and data mining (WSDM 2010)* (pp. 31–40), New York.

Berger, A., & Lafferty, J. (1999). Information retrieval as statistical translation. In *Proc. 22nd ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 222–229).

Bergsma, S., & Wang, Q. I. (2007). Learning noun phrase query segmentation. In *Proc. of EMNLP-CoNLL.*

Blei, D., Griffiths, T., Jordan, M., & Tenenbaum, J. (2003a). Hierarchical topic models and the nested Chinese restaurant process. In *Proc. 16th conf. of advances in neural information processing systems*.

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8, © Springer-Verlag Berlin Heidelberg 2011

Blei, D., Ng, A., & Jordan, M. (2003b). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

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

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum*, *36*(2), 3–10.

Broder, A., Ciccolo, P., Gabrilovich, E., Josifovski, V., Metzler, D., Riedel, L., & Yuan, J. (2009). Online expansion of rare queries for sponsored search. In *Proceedings of the 18th international conference on World Wide Web*, WWW '09 (pp. 511–520). New York: ACM.

Buckley, C. (2004). Why current IR engines fail. In *Proc. 27th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 584–585).

Buckley, C., & Salton, G. (1995). Optimization of relevance feedback weights. In *Proc. 18th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 351–357).

Buckley, C., Dimmick, D., Soboroff, I., & Voorhees, E. (2006). Bias and the limits of pooling. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 619–620).

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *Proc. 22nd intl. conference on machine learning* (pp. 89–96).

Burges, C. J. C., Ragno, R., & Le, Q. V. (2007). Learning to rank with nonsmooth cost functions. In *Proc. 19th conf. of advances in neural information processing systems* (pp. 193–200).

Buttcher, S., & Clarke, C. (2005). Efficiency vs. effectiveness in terabyte-scale information retrieval. In *Proc. 13th intl. conf. on World Wide Web*.

Büttcher, S., Clarke, C. L. A., & Lushman, B. (2006a). Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 621–622).

Büttcher, S., Clarke, C. L. A, & Soboroff, I. (2006b). The TREC 2006 terabyte track. In *Proc. 15th text retrieval conference*.

Cao, G., Nie, J.-Y., Gao, J., & Robertson, S. (2008). Selecting good expansion terms for pseudo-relevance feedback. In *Proc. 31st ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M. Maarek, Y. S., & Soffer, A. (2001). Static index pruning for information retrieval systems. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, SIGIR '01 (pp. 43–50). New York: ACM.

Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, *14*(3), 462–467.

Clarke, C., Cormack, G., & Burkowski, F. (1995). Shortest substring ranking (MultiText experiments for TREC-4). In *Proc. 4th intl. conf. on World Wide Web*.

Clarke, C., Scholar, F., & Soboroff, I. (2006). Overview of the TREC 2005 terabyte track. In *Proc. 14th intl. conf. on World Wide Web*.

Clarke, C. L. A., & Cormack, G. V. (2000). Shortest-substring retrieval and ranking. *ACM Transactions on Information Systems*, *18*(1), 44–78.

Collins-Thompson, K., & Callan, J. (2005). Query expansion using random walk models. In *Proc. 14th intl. conf. on information and knowledge management* (pp. 704–711).

Cooper, W. S. (1991). Some inconsistencies and misnomers in probabilistic information retrieval. In *Proc. 14th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 57–61).

Craswell, N., de Vries, A. P., & Soboroff, I. (2005a). Overview of the TREC 2005 enterprise track. In *Proc. 14th intl. conf. on World Wide Web*.

Craswell, N., Robertson, S., Zaragoza, H., & Taylor, M. (2005b). Relevance weighting for query independent evidence. In *Proc. 28th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 416–423).

Croft, W. B. (1986). Boolean queries and term dependencies in probabilistic retrieval models. *Journal of the American Society for Information Science*, *37*(4), 71–77.

Croft, W. B., & Harper, D. (1979). Using probabilistic models of information retrieval without relevance information. *Journal of Documentation*, *35*(4), 285–295.

Croft, W. B., Turtle, H., & Lewis, D. (1991). The use of phrases and structured queries in information retrieval. In *Proc. 14th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 32–45).

Cummins, R., & O'Riordan, C. (2009). Learning in a pairwise term-term proximity framework for information retrieval. In *Proc. 32nd ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

de Kretser, O., & Moffat, A. (1999). Effective document presentation with a locality-based similarity heuristic. In *Proc. 22nd ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 113–120).

Deerwester, S., Dumais, S., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Science*, *41*(6), 391–407.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Diaz, F. (2005). Regularizing ad hoc retrieval scores. In *Proc. 14th intl. conf. on information and knowledge management* (pp. 672–679).

Diaz, F., & Metzler, D. (2006). Improving the estimation of relevance models using large external corpora. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 154–161).

Eguchi, K. (2005). NTCIR-5 query expansion experiments using term dependence models. In *Proc. of the fifth NTCIR workshop meeting on evaluation of information access technologies* (pp. 494–501).

Fagan, J. (1987). Automatic phrase indexing for document retrieval: An examination of syntactic and non-syntactic methods. In *Proc. tenth ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 91–101).

Fan, W., Gordon, M. D., & Pathak, P. (2004). A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing & Management*, *40*(4), 587–602.

Fang, H., & Zhai, C. (2005). An exploration of axiomatic approaches to information retrieval. In *Proc. 28th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 480–487).

Fang, H., & Zhai, C. (2006). Semantic term matching in axiomatic approaches to information retrieval. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 115–122).

Fox, E. A., & Shaw, J. A. (1993). Combination of multiple searches. In *Proc. 2nd intl. conf. on World Wide Web* (pp. 243–252).

Gao, J., Nie, J., Wu, G., & Cao, G. (2004). Dependence language model for information retrieval. In *Proc. 27th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 170–177).

Gao, J., Qi, H., Xia, X., & Nie, J. (2005). Linear discriminant model for information retrieval. In *Proc. 28th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 290–297).

Gey, F. (1994). Inferring probability of relevance using the method of logistic regression. In *Proc. 17th ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

Griffiths, T. L., Steyvers, M., Blei, D. M., & Tenenbaum, J. B. (2005). Integrating topics and syntax. In *Proc. 17th conf. of advances in neural information processing systems* (pp. 537–544).

Guo, J., Xu, G., Li, H., & Cheng, X. (2008). A unified and discriminative model for query refinement. In *Proc. of 31st ann. intl. ACM SIGIR conf. on research and development in information retrieval*. New York: ACM.

Harman, D. (1993). Overview of the first TREC conference. In *Proc. 16th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 36–47).

Harper, D., & van Rijsbergen, C. J. (1978). An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation*, *34*(3), 189–216.

Harter, S. P. (1975). A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Science*, *26*(5), 197–206 and 280–289.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proc. 22nd ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 50–57).

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, *20*(4), 422–446.

Ji, X. & Zha, H. (2003). Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval*, SIGIR '03 (pp. 322–329). New York: ACM.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proc. 8th ann. intl. ACM SIGKDD conf. on knowledge discovery and data mining* (pp. 133–142).

Joachims, T. (2005). A support vector method for multivariate performance measures. In *Proc. 22nd intl. conference on machine learning* (pp. 377–384).

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, *46*(5), 604–632.

Kraaij, W., Westerveld, T., & Hiemstra, D. (2002). The importance of prior probabilities for entry page search. In *Proc. 25th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 27–34).

Kumaran, G., & Carvalho, V. R. (2009). Reducing long queries using query quality predictors. In *Proc. 32nd ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

Kurland, O., & Lee, L. (2004). Corpus structure, language models, and ad hoc information retrieval. In *Proc. 27th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 194–201).

Lang, H., Metzler, D., Wang, B., & Li, J.-T. (2010). Improved latent concept expansion using hierarchical Markov random fields. In *Proc. 19th intl. conf. on information and knowledge management*, CIKM '10 (pp. 249–258). New York: ACM.

Lavrenko, V. (2004). *A generative theory of relevance*. PhD thesis, University of Massachusetts, Amherst, MA.

Lavrenko, V., & Croft, W. B. (2001). Relevance-based language models. In *Proc. 24th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 120–127).

Le, Q., & Smola, A. (2007). Direct optimization of ranking measures. http://www.citebase.org/abstract?id=oai:arXiv.org:0704.3359.

Lease, M. (2009). An improved Markov random field model for supporting verbose queries. In *Proc. 32nd ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

Lease, M., Croft, W. B., & Allan, J. (2009). Regression rank: Learning to meet the opportunity of descriptive queries. In *Proc. 31st European conf. on information retrieval*.

Lin, J., Metzler, D., Elsayed, T., & Wang, L. (2009). Of ivory and smurfs: Loxodontan mapreduce experiments for web search. In *Proc. 18th intl. conf. on World Wide Web*.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, *3*(3).

Liu, X., & Croft, W. B. (2004). Cluster-based retrieval using language models. In *Proc. 27th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 186–193).

Losee, R. Jr. (1994). Term dependence: truncating the Bahadur Lazarsfeld expansion. *Information Processing & Management*, *30*(2), 293–303.

Macdonald, C. & Ounis, I. (2007). Expertise drift and query expansion in expert search. In *Proceedings of the sixteenth ACM conference on information and knowledge management*, CIKM '07 (pp. 341–350). New York: ACM.

Matveeva, I., Burges, C., Burkard, T., Laucius, A., & Wong, L. (2006). High accuracy retrieval with multiple nested ranker. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 437–444).

McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *Proc. 19th conf. on uncertainty in artificial intelligence*.

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

Metzler, D., & Croft, W. B. (2005). A Markov random field model for term dependencies. In *Proc. 28th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 472–479).

Metzler, D., & Croft, W. B. (2007). Latent concept expansion using Markov random fields. In *Proc. 30th ann. intl. ACM SIGIR conf. on research and development in information retrieval*.

Metzler, D., & Manmatha, R. (2004). An inference network approach to image retrieval. In *Proc. 3rd intl. conf. on image and video retrieval* (pp. 42–50).

Metzler, D., Lavrenko, V., & Croft, W. B. (2004a). Formal multiple Bernoulli models for language modeling. In *Proc. 27th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 540–541).

Metzler, D., Strohman, T., Turtle, H., & Croft, W. B. (2004b). Indri at TREC 2004: Terabyte track. In *Proc. 13th intl. conf. on World Wide Web*.

Metzler, D., Diaz, F., Strohman, T., & Croft, W. B. (2005a). UMass robust 2005: Using mixtures of relevance models for query expansion. In *Proc. 14th intl. conf. on World Wide Web*.

Metzler, D., Strohman, T., Zhou, Y., & Croft, W. B. (2005b). Indri at TREC 2005: terabyte track. In *Proc. 14th intl. conf. on World Wide Web*.

Metzler, D., Strohman, T., & Croft, W. B. (2006). Lessons learned from three terabyte tracks. In *Proc. 15th intl. conf. on World Wide Web*.

Metzler, D., Dumais, S., & Meek, C. (2007). Similarity measures for short segments of text. In *Proc. 29th European conf. on information retrieval* (pp. 16–27).

Mishne, G., & de Rijke, M. (2005). Boosting web retrieval through query operations. In *Proc. 27th European conf. on information retrieval* (pp. 502–516).

Morgan, W., Greiff, W., & Henderson, J. (2004). *Direct maximization of average precision by hill-climbing with a comparison to a maximum entropy approach* (Technical report). MITRE.

Morik, K., Brockhausen, P., & Joachims, T. (1999). Combining statistical learning with a knowledge-based approach—a case study in intensive care monitoring. In *Proc. 16th intl. conference on machine learning*.

Murdock, V., & Croft, W. B. (2005). A translation model for sentence retrieval. In *Proc. HLT '05* (pp. 684–691). Morristown: Association for Computational Linguistics.

Nallapati, R. (2004). Discriminative models for information retrieval. In *Proc. 27th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 64–71).

Nallapati, R., & Allan, J. (2002). Capturing term dependencies using a language model based on sentence trees. In *Proc. 11th intl. conf. on information and knowledge management* (pp. 383–390).

Ntoulas, A., & Cho, J. (2007). Pruning policies for two-tiered inverted index with correctness guarantee. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval*, SIGIR '07 (pp. 191–198). New York: ACM.

Ogilvie, P., & Callan, J. (2003). Combining document representations for known-item search. In *Proc. 26th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 143–150).

Ounis, I., de Rijke, M., MacDonald, C., Mishne, G., & Soboroff, I. (2006). Overview of the TREC-2006 blog track. In *Proc. 15th intl. conf. on World Wide Web*.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 conference on empirical methods in natural language processing (EMNLP)* (pp. 79–86).

Papka, R., & Allan, J. (1997). *Why bigger windows are better than smaller ones* (Technical report). University of Massachusetts, Amherst.

Peng, J., Macdonald, C., He, B., Plachouras, V., & Ounis, I. (2007). Incorporating term dependency in the dfr framework. In *Proc 30th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 843–844).

Peng, Y., & He, D. (2006). Direct comparison of commercial and academic retrieval system: an initial study. In *Proc. 15th intl. conf. on information and knowledge management* (pp. 806–807).

Pickens, J., & Croft, W. B. (1999). An exploratory analysis of phrases in text retrieval. In *Proc. of RIAO 2000*.

Ponte, J., & Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proc. 21st ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 275–281).

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: the art of scientific computing*. Cambridge: Cambridge University Press.

Robertson, S. (1977). The probability ranking principle in IR. *Journal of Documentation*, *33*(4), 294–303.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, *60*(5), 503–520.

Robertson, S., & Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proc. 17th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 232–241).

Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1994). Okapi at TREC-3. In *Proc. 3rd intl. conf. on World Wide Web* (pp. 109–126).

Robertson, S., Zaragoza, H., & Taylor, M. (2004). Simple bm25 extension to multiple weighted fields. In *Proc. 13th intl. conf. on information and knowledge management* (pp. 42–49).

Robertson, S. E., & Spärck Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, *27*(3), 129–146.

Robertson, S. E., & Walker, S. (1997). On relevance weights with little relevance information. In *Proc. 20th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 16–24).

Robertson, S. E., van Rijsbergen, C. J., & Porter, M. F. (1980). Probabilistic models of indexing and searching. In *Proc. 3rd ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 35–56).

Rocchio, J. J. (1971). *Relevance feedback in information retrieval* (pp. 313–323). New York: Prentice-Hall.

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, *88*(8), 1270–1278.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523.

Shen, X., & Zhai, C. (2005). Active feedback in ad hoc information retrieval. In *Proc. 28th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 59–66).

Si, L., & Callan, J. (2001). A statistical model for scientific readability. In *Proc. 10th intl. conf. on information and knowledge management* (pp. 574–576).

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. In *Proc. 19th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 21–29).

Song, F., & Croft, W. B. (1999). A general language model for information retrieval. In *Proc. 8th intl. conf. on information and knowledge management* (pp. 316–321).

Spärck Jones, K. (1971). *Automatic keyword classification for information retrieval*. Stoneham: Butterworths.

Spärck Jones, K. (2005). *Wearing proper combinations* (Technical report). University of Cambridge.

Spärck Jones, K. & van Rijsbergen, C. J. (1976). Information retrieval test collections. *Journal of Documentation*, *32*(1), 59–72.

Srikanth, M., & Srihari, R. (2002). Biterm language models for document retrieval. In *Proc. 25th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 425–426).

Strohman, T., & Croft, W. B. (2007). Efficient document retrieval in main memory. In *Proc 30th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 175–182).

Strohman, T., Metzler, D., Turtle, H., & Croft, W. B. (2004). Indri: A language model-based search engine for complex queries. In *Proceedings of the international conference on intelligence analysis*.

Strohman, T., Turtle, H., & Croft, W. B. (2005). Optimization strategies for complex queries. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*, SIGIR '05 (pp. 219–225). New York: ACM.

Tan, B., & Peng, F. (2008). Unsupervised query segmentation using generative language models and wikipedia. In *Proc. 17th intl. World Wide Web conf.* New York: ACM.

Tao, T., Wang, X., Mei, Q., & Zhai, C. (2006). Language model information retrieval with document expansion. In *Proc. of HLT/NAACL* (pp. 407–414).

Tao, T. & Zhai, C. X. (2007). An exploration of proximity measures in information retrieval. In *Proc. 30th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 295–302). New York: ACM.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, *58*, 267–288.

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

van Rijsbergen, C. J. (1977). A theoretical basis for the use of cooccurrence data in information retrieval. *Journal of Documentation*, *33*(2), 106–119.

Vechtomova, O., Karamuftuoglu, M., & Robertson, S. E. (2006). On document relevance and lexical cohesion between query terms. *Information Processing & Management*, *42*(5), 1230–1247.

Voorhees, E. (1999). The TREC-8 question answering track report. In *Proc. 8th intl. conf. on World Wide Web* (pp. 77–82).

Voorhees, E. (2004). Overview of the TREC 2004 robust retrieval track. In *Proc. 13th intl. conf. on World Wide Web*.

Voorhees, E. (2005). Overview of the TREC 2005 robust retrieval track. In *Proc. 14th intl. conf. on World Wide Web*.

Wang, L., Lin, J., & Metzler, D. (2010a). Learning to efficiently rank. In *Proc. 33rd ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 138–145). Geneva, Switzerland.

Wang, L., Metzler, D., & Lin, J. (2010b). Ranking under temporal constraints. In *Proc. 19th intl. conf. on information and knowledge management*, Toronto, Canada.

Wei, X., & Croft, W. B. (2006). LDA-based document models for ad-hoc retrieval. In *Proc. 29th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 178–185).

Wei, X., & Croft, W. B. (2007). Modeling term associations for ad-hoc retrieval performance within language modeling framework. In *Proc. 29th European conf. on information retrieval* (pp. 52–63).

Wong, S. K. M., & Yao, Y. Y. (1988). Linear structure in information retrieval. In *Proc. 11th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 219–232).

Xu, J., & Croft, W. B. (2000). Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, *18*(1), 79–112.

Yu, C. T., Buckley, C., Lam, K., & Salton, G. (1983). *A generalized term dependence model in information retrieval* (Technical report). Cornell University.

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. In *Proc. 30th ann. intl. ACM SIGIR conf. on research and development in information retrieval*, to appear.

Zhai, C., & Lafferty, J. (2001a). Model-based feedback in the language modeling approach to information retrieval. In *Proc. 10th intl. conf. on information and knowledge management* (pp. 403–410).

Zhai, C., & Lafferty, J. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proc. 24th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 334–342).

Zhai, C., & Lafferty, J. (2002). Two-stage language models for information retrieval. In *Proc. 25th ann. intl. ACM SIGIR conf. on research and development in information retrieval* (pp. 49–56).

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

Zhou, Y., & Croft, W. B. (2005). Document quality models for web ad hoc retrieval. In *Proc. 14th intl. conf. on information and knowledge management* (pp. 331–332).

Index

D. Metzler, *A Feature-Centric View of Information Retrieval*,
The Information Retrieval Series 27,
DOI 10.1007/978-3-642-22898-8, © Springer-Verlag Berlin Heidelberg 2011