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.

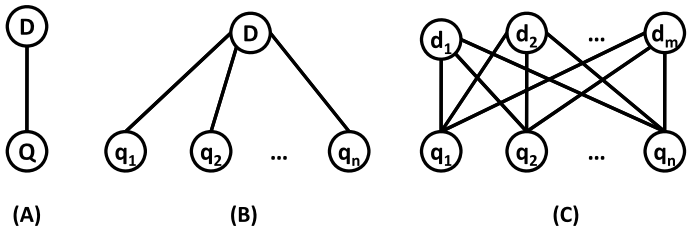

Fig. 3.1 Three possible ways to model the joint distribution $P(Q, D)$ using Markov random fields

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.

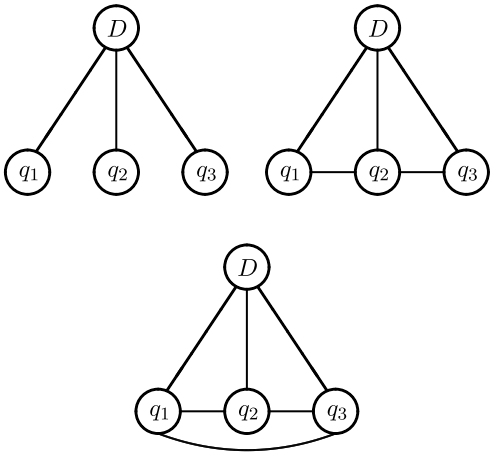

Fig. 3.2 Example Markov random field model for three query terms under various independence assumptions, including full independence (*top left*), sequential dependence (*top right*), and full dependence (*bottom*)

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

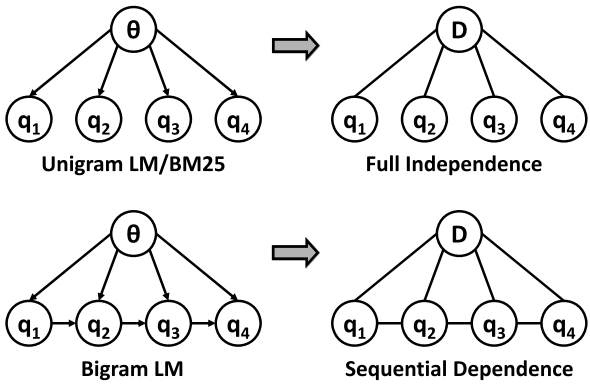

Fig. 3.3 Illustration showing how the full independence model generalizes unigram language modeling and BM25 (*top*), and how the sequential dependence model generalizes bigram language modeling (*bottom*)

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

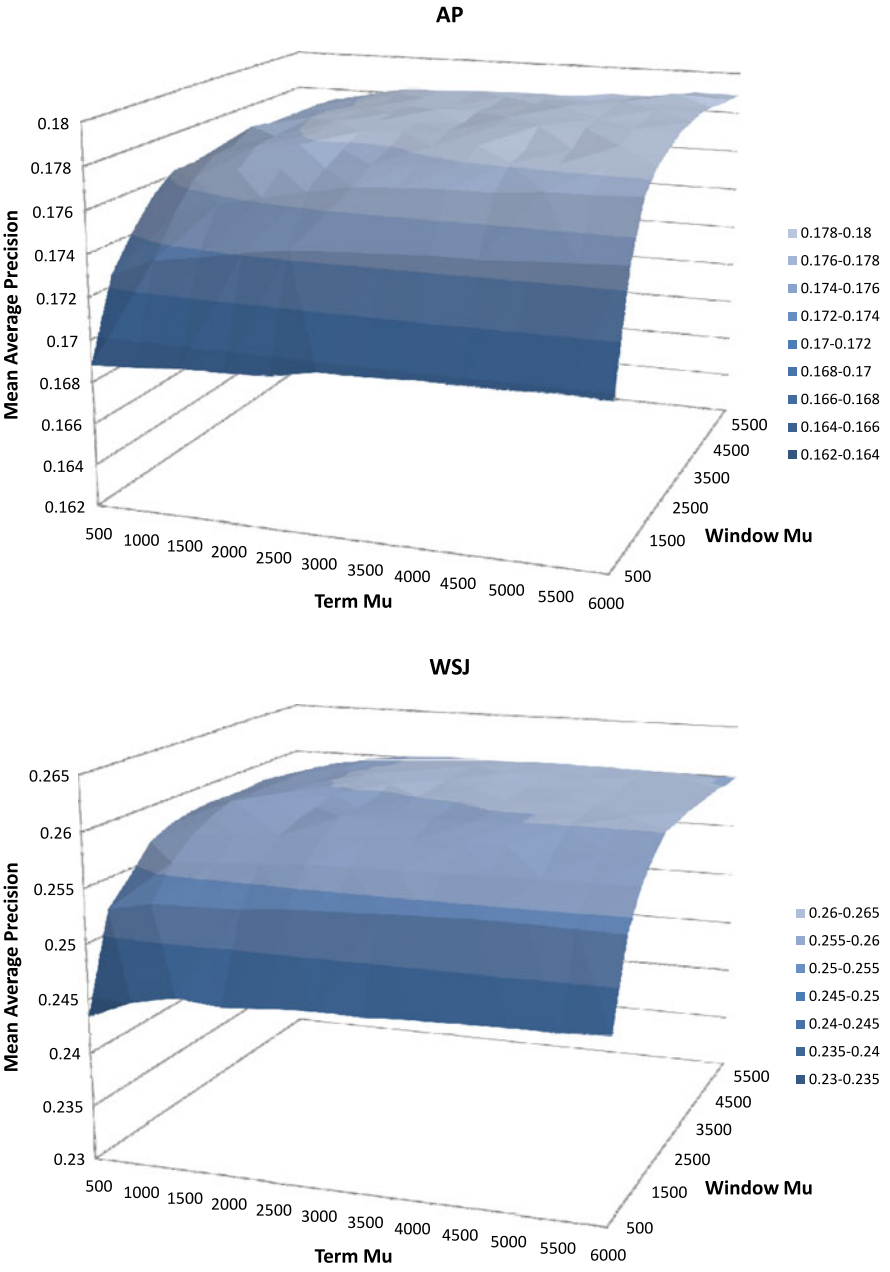

Fig. 3.5 Training set mean average precision as a function of term and window smoothing parameters using the sequential dependence model on the AP, WSJ, ROBUST04, and WT01G data sets (*top to bottom*)

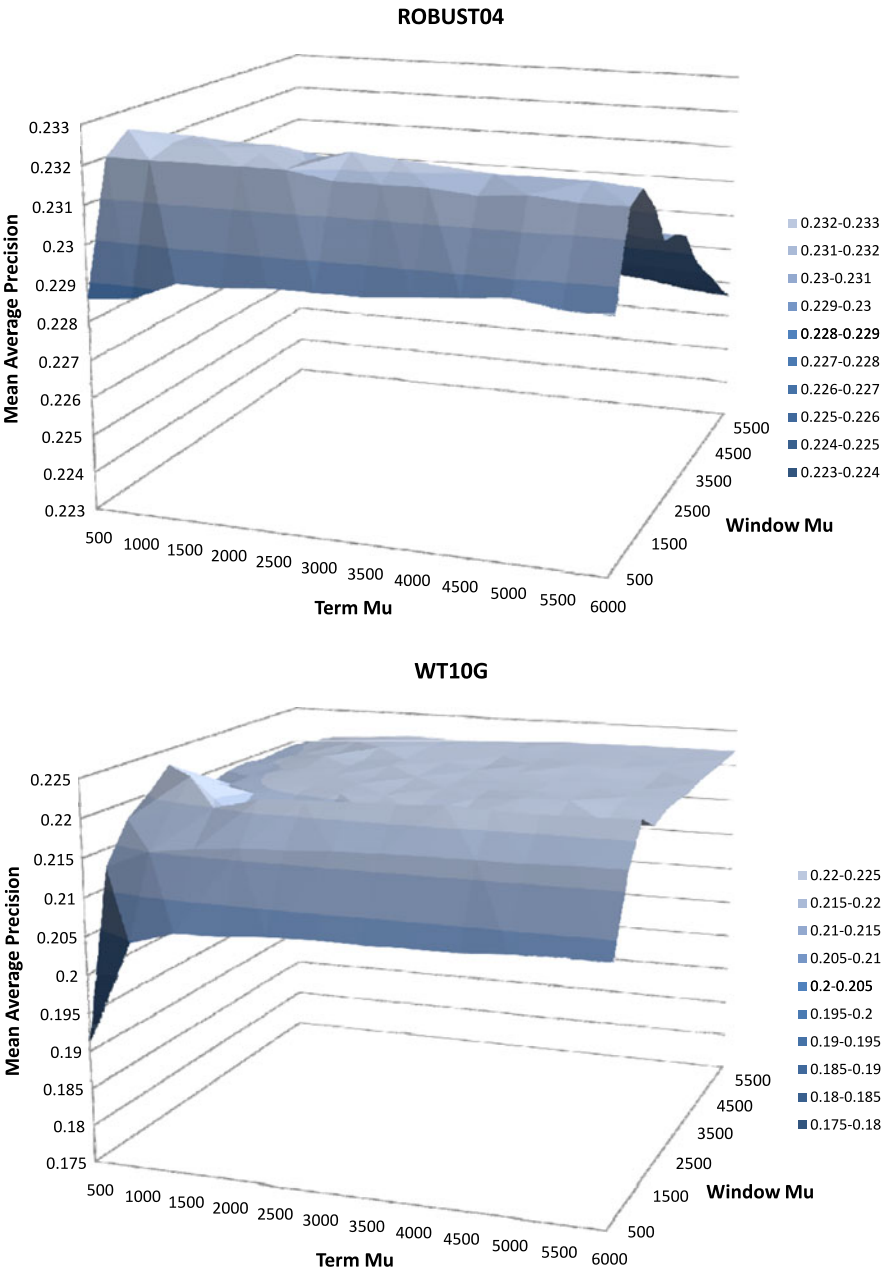

Fig. 3.5 (Continued)

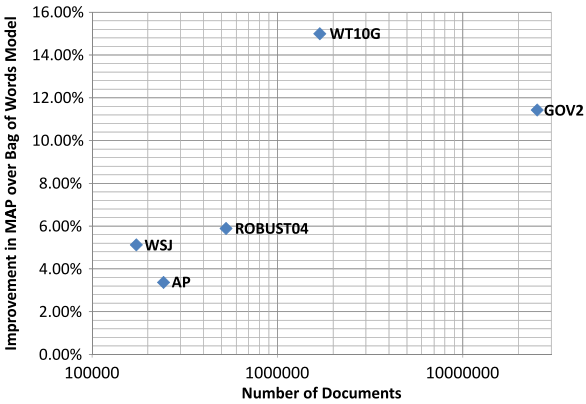

Fig. 3.6 Relationship between the number of documents in a collection and the relative improvement in mean average precision of the MRF-FD model over unigram language modeling (MRF-FI). Note that the x-axis is log scaled

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

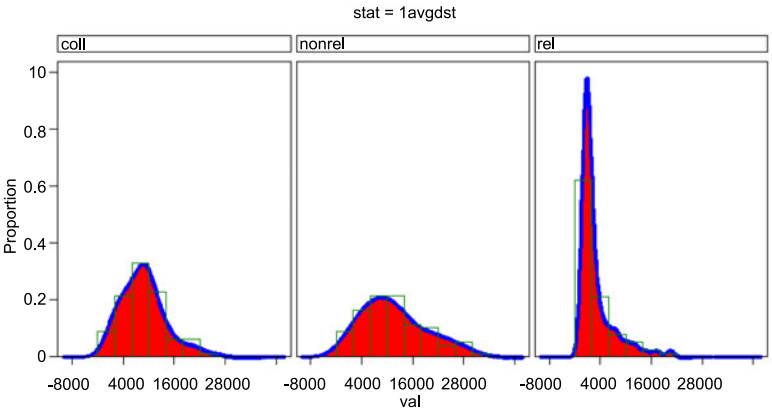

Fig. 3.7 Plot of average distance between adjacent query terms for WT10G data set. *Coll* represents the entire collection, *nonrel* the set of judged non-relevant documents, and *rel* the set of judged relevant documents

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).

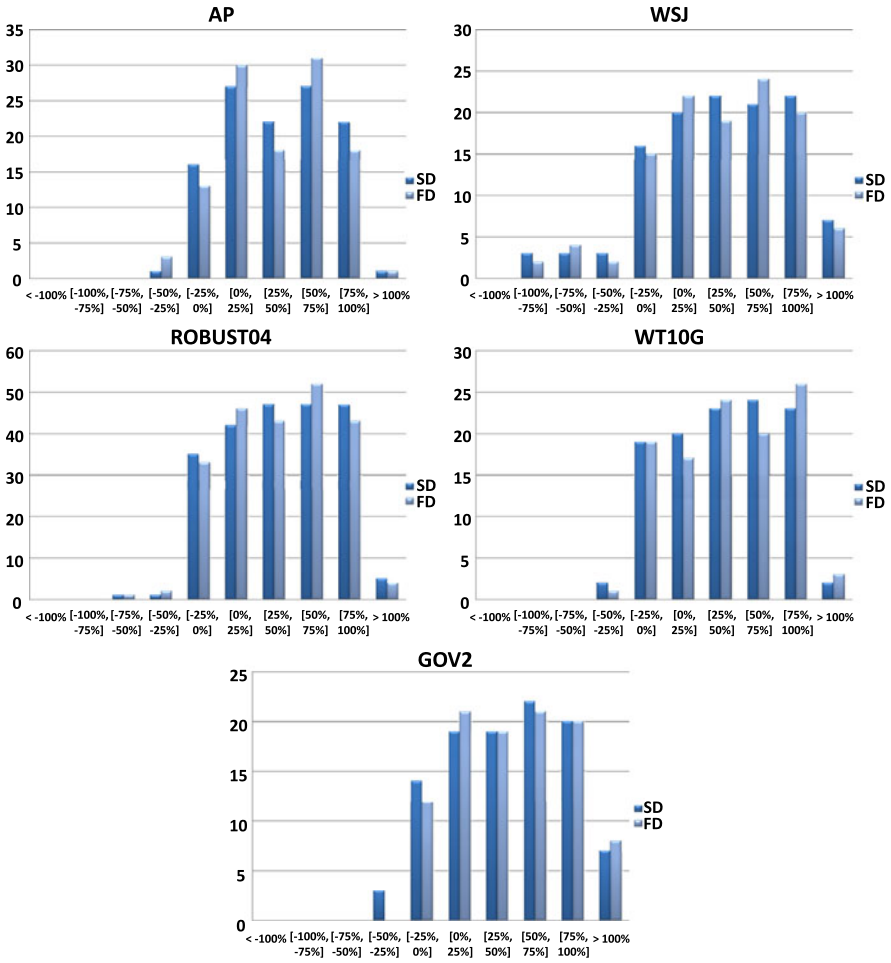

Fig. 3.8 Robustness of MRF-SD and MRF-FD models on the AP, WSJ, ROBUST04, WT10G, and GOV2 test sets. The MRF-FI model is used as the baseline by which the improvements are computed. The evaluation metric used is average precision

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:

$$\begin{aligned}
P(D|Q) \stackrel{\text{rank}}{=} & \lambda_{T_D} \sum_{(q_i,D)\in T_{QD}} \frac{(k_1^t+1)tf_{w,D}}{k_1^t\left((1-b^t)+b^t\frac{|D|}{|D|_{\text{avg}}}\right)+tf_{w,D}} idf(w) \\
& + \lambda_{O_D} \sum_{(q_1,q_2,D)\in O_{QD}} \frac{(k_1^w+1)tf_{\#1(q_1q_2),D}}{k_1^w\left((1-b^w)+b^w\frac{|D|}{|D|_{\text{avg}}}\right)+tf_{\#1(q_1q_2),D}} \\
& \times idf\left(\#1(q_2q_2)\right) \\
& + \lambda_{U_D} \sum_{(q_1,q_2,D)\in U_{QD}} \frac{(k_1^w+1)tf_{\#\text{uw8}(q_1q_2),D}}{k_1^w\left((1-b^w)+b^w\frac{|D|}{|D|_{\text{avg}}}\right)+tf_{\#\text{uw8}(q_1q_2),D}} \\
& \times idf\left(\#\text{uw8}(q_1q_2)\right)
\end{aligned} \tag{3.27}$$

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

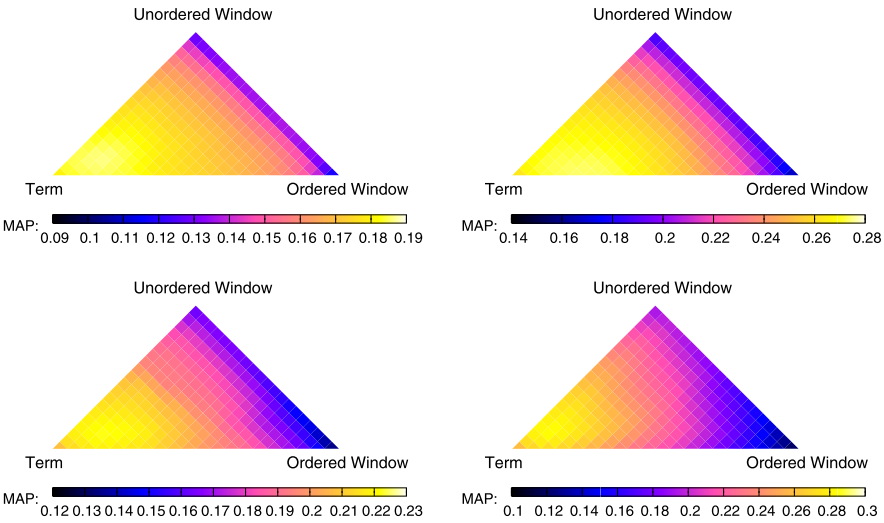

Fig. 3.9 Mean average precision values plotted over MRF-SD parameter simplex for AP, WSJ, WT10g, and GOV2 collections

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.

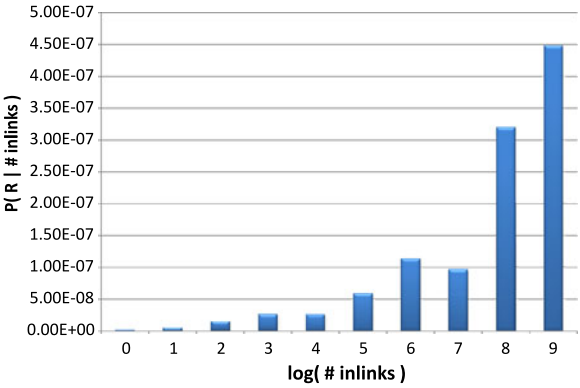

Fig. 3.11 Inlink count prior

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

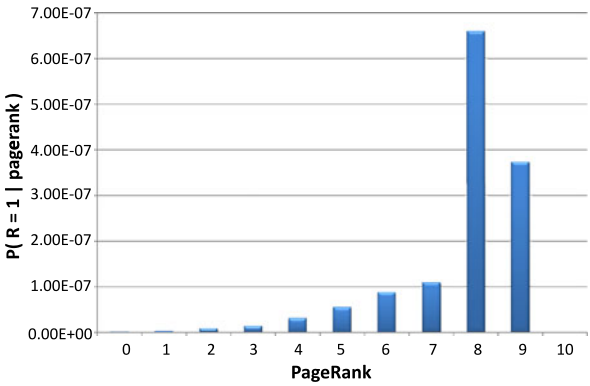

Fig. 3.12 PageRank prior

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%

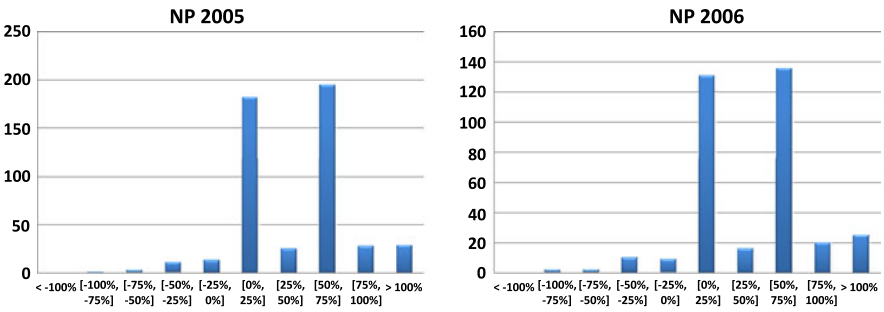

Fig. 3.13 Robustness of the MRF-NP models for the 2005 and 2006 Terabyte Track named page finding data sets. The LM-Mixture model is used as the baseline by which the improvements were computed. The evaluation metric used is reciprocal rank

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.