Risk Minimization and Language Modeling in Text Retrieval

ChengXiang Zhai

July 2002

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Thesis Committee:
John Lafferty, Chair
Jamie Callan
Jaime Carbonell
David A. Evans, Clairvoyance Corporation
W. Bruce Croft, University of Massachusetts, Amherst

This research was sponsored in part by the Advanced Research and Development Activity in Information Technology (ARDA) under its Statistical Language Modeling for Information Retrieval Research Program. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, DARPA, the U.S. government or any other entity.

Keywords: information retrieval, text retrieval, risk minimization, Bayesian decision theory, language modeling

To Yan

Acknowledgments

I wish to express my greatest thanks to my advisor, John Lafferty. John's excellent guidance has been absolutely essential for the completion of this thesis. From the very beginning, he has treated me as a peer and a friend, and given me the right amount of freedom and guidance. It has been such a joy to work with him. Over the years, I have learned many many things from him – I learned to truly appreciate high quality in research, I learned many technical skills for developing and applying statistical language models, and I even learned from him how to improve my technical writing. I am grateful to John for his extremely constructive technical guidance, especially his direct contribution to the formulation of the retrieval problem as a statistical risk minimization framework, which is the basis of the whole thesis. John's high research standard and principled methodology have influenced my research philosophy fundamentally.

I have many reasons for being grateful to David A. Evans, who was my initial Ph.D. thesis advisor and the first person to inspire my initial interest in the exciting research areas of natural language processing and information retrieval. David supervised my early research work in these areas, including a Master thesis on exploring the use of different levels of noun phrases for document indexing, and has been a mentor since the very beginning of my graduate study. His unique insights in identifying important IR problems and forming practical solutions have deeply influenced my way of looking for realistic solutions to real world problems. David also provided me with opportunities for working with real world IR systems and participating in TREC, through sponsoring my internships and regular employment at Clairvoyance Corporation (formerly Claritech Corporation). Through these industrial research experiences, I learned the importance of users and solid evaluation, which are reflected in this thesis. Moreover, David served as a member on my thesis committee and provided many valuable comments.

I am also grateful to the other members of my committee, Jamie Callan, Jaime Carbonell, and W. Bruce Croft for their constructive feedback about the thesis. Jamie has been very helpful ever since the time of the thesis proposal. He has provided very useful comments that helped me to plan my thesis research more realistically. He also gave me many concrete comments that have made the evaluation in my thesis much more solid. Jaime gave me very insightful comments on the risk minimization framework and specific suggestions for exploring the Maximal Marginal Relevance approaches in the language modeling framework. I am grateful to Bruce for pioneering and encouraging research on

applying statistical language models to IR. Discussions with him were very helpful for validating the research ideas developed in the thesis.

Thanks to the many people with whom I have had useful discussions of ideas in my thesis. I especially would like to mention Stephen Robertson, William Cohen, Victor Lavrenko, Rong Jin, Yi Zhang, Roni Rosenfeld, Tom Minka, Jerry Zhu, Paul Ogilvie, Luo Si, Kevyn Collins-Thompson, and Bob Carpenter. Thanks to Lauri Lafferty for carefully proofreading this thesis, which helped to improve the presentation significantly.

I am grateful to all the friends, fellow students, and colleagues with whom I spent my time as a graduate student at Carnegie Mellon University. In particular, thanks to Xiang Tong for being my officemate for more than seven years both at Carnegie Mellon University and at Clairvoyance Corporation. Xiang and I shared a lot of good times discussing virtually every aspect of life and research. Special thanks to Susi Burger for being a wonderful officemate during the last two years of my Ph.D. study and for lending me a desktop initially while I was waiting for my own desktop to come. Thanks to Thi Nhu Truong for being a great teammate to work with on the Lemur project. We had a great time together debugging and releasing the Lemur toolkit. Thanks to Yiming Yang, Guy Lebanon, Bing Zhao, Judith Klein-Seetharaman, Xinghua Lu, Emilia Stoica, Yan Qu, and Klaus Zechner for many stimulating technical discussions.

I owe a great deal to my parents and my brother for their love and support without which my achievements would not have been possible. Thanks to my parents for fostering my desire to pursue scientific inquiry, my persistence in working, and a confident and optimistic life philosophy. Thanks to my brother for his tremendous help both financially and spiritually to make my transition from China to the US as smooth as it could be.

I am deeply indebted to my dear wife Yan for her love and strong support for my graduate study. Yan has influenced my life very positively and fundamentally ever since we met. She is always appreciative of high quality, which has stimulated me to set my goals high. I am grateful to her for encouraging me to pursue a second Ph.D. degree at Carnegie Mellon University, and for her continuing support throughout my long journey to finish it. This thesis is dedicated to her.

My acknowledgments would not be complete without also thanking our sons Alex and David for adding so much pleasure to my life and for pushing me in a positive way to finish the thesis.

Abstract

With the dramatic increase in online information in recent years, text retrieval is becoming increasingly important. It has always been a significant scientific challenge to develop principled retrieval approaches that also perform well empirically, since, so far, those theoretically well-motivated models have rarely led directly to good performance. It is also a great challenge to develop models that go beyond the traditional notion of topical relevance and capture more user factors, such as topical redundancy and sub-topic diversity.

This thesis presents a new text retrieval framework based on Bayesian decision theory. The framework unifies several existing retrieval models, including the recently proposed language modeling approach, within one general probabilistic framework. It also facilitates the development of new principled approaches to text retrieval with the potential of going beyond the traditional notion of topical relevance. In this framework, queries and documents are modeled using statistical language models (i.e., probabilistic models of text), user preferences are modeled through loss functions, and retrieval is cast as a risk minimization problem.

While traditional retrieval models rely heavily on ad hoc parameter tuning to achieve satisfactory retrieval performance, the use of language models in the risk minimization framework makes it possible to exploit statistical estimation methods to improve retrieval performance and set retrieval parameters automatically. As a special case of the framework, we present a two-stage language model that, according to extensive evaluation, achieves excellent retrieval performance without any ad hoc parameter tuning.

The use of language models in retrieval also provides more guidance on how to improve retrieval performance (e.g., through using more reasonable language models and/or more accurate estimation methods). As another special case of the risk minimization framework, we derive a Kullback-Leibler divergence retrieval model that can exploit feedback documents to improve the estimation of query models. Feedback has so far been dealt with heuristically in the language modeling approach to retrieval. The KL-divergence model provides a more principled way of performing feedback by treating it as query model updating. We propose two specific query model updating algorithms based on feedback documents. Evaluation indicates that both algorithms are effective for feedback, and the updated query models outperform the original

query models significantly.

The risk minimization retrieval framework further allows for incorporating user factors beyond the traditional notion of topical relevance. We present language models that can capture redundancy and sub-topics in documents, and study loss functions that can rank documents in terms of both relevance and sub-topic diversity. Evaluation shows that the proposed language models can effectively capture redundancy and can outperform the relevance-based ranking method for the aspect retrieval task.

The risk minimization framework opens up many new possibilities for developing principled approaches to text retrieval, and makes it possible to explore different retrieval models more systematically, through considering different loss functions and using different language models in the framework.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent years have seen explosive growth in the volume of information. For example, from December 1997 to February 1999, the Web more than doubled in size, from 320 million pages to 800 million, encompassing about 15 terabytes of information (Lawrence and Giles, 1999). And, as of the year 2001, the Google index has more than 3 billion documents, of which 2 billion are web pages (Sherman, 2001). Other types of textual information, such as books, newspapers, and periodicals are also growing rapidly. According to a recent study at Berkeley (Lyman and Varian, 2000), the worldwide production of original content, stored digitally using standard compression methods, is at least 1 terabyte/year for books, 2 terabytes/year for newspapers, 1 terabyte/year for periodicals, and 19 terabytes/year for office documents.

The high volume of information makes effectively and efficiently managing the online information a significant challenge, and causes *information overload*, i.e., the inability to extract needed knowledge from an immense quantity of information (Wurman, 1989). Information retrieval is by far the most useful technique to address the problem of information overload. The retrieval of textual information (i.e., text retrieval) is especially important, because the most frequently wanted information is often textual, and techniques for retrieving textual information can be useful for retrieving other media information, when there is companion text.

Given a document collection (i.e., a set of unordered text documents), the task of Text Retrieval (TR) can be defined as using a user query (i.e., a description of the user's information need) to identify a subset of documents that can satisfy the user's information

need. [1] A major difficulty in TR is to accurately judge whether a particular set of documents would satisfy a user's information need. The criterion is inherently impossible to formalize, because it is generally very hard for a user to prescribe the exact information need completely and precisely through a query, and even more difficult for a computer to understand the meaning of a query precisely. Thus, TR is really not a well-defined task, in the sense that the correctness of the solution to the retrieval problem can only be evaluated by a user empirically.

Traditionally, the documents that can satisfy a user's information need are called *relevant* documents, and thus, the retrieval task is to find all relevant documents for a given query. The notion of "relevance" is very complicated. Like a user's information need, relevance is generally imprecise and depends on the situation or context of the retrieval task. Two assumptions are often made to simplify the retrieval task. First, the relevance of one document is assumed to be independent of other documents, including those already retrieved (*independent relevance assumption*). Second, the relevance of a document is assumed to mean the level of *topical* relevance of the document with respect to the query (*topical relevance assumption*). Under these two constraints, the retrieval task is essentially to evaluate the *topical* relevance value of each document *independently* with respect to a query, which makes the retrieval task more tractable. In reality, however, neither of these two assumptions can be expected to hold. For example, a user is unlikely to be interested in seeing duplicates of the same document, even if the duplicates are all topically relevant to the query. This means that the utility of retrieving a document at least depends on the previously retrieved documents that the user has already seen. Also, topical relevance is only one of the many attributes of a document that a user may care about. For example, if the user is a child, the readability of the documents would matter. All of this suggests that it is highly desirable to develop retrieval models that can go beyond the notion of *independent topical relevance*. Indeed, the need to go beyond topical relevance has already been well-recognized in the literature (Saracevic, 1970; Froehlich, 1994). There is also some work on ranking documents by considering redundancy (Carbonell and Goldstein, 1998), but such "non-topical" factors have only been considered on top of an existing retrieval model. It is still fair to say that most existing retrieval models have focused on only topical relevance. From now on, unless explicitly mentioned, we will use relevance to mean topical relevance.

[1] A larger scope of IR or TR is often taken to include not only the narrow sense of retrieval, but also routing, categorization, and clustering (Lewis, 1992). In this thesis, we define IR as retrieving documents from a collection that satisfy a user's information need, which is also referred to as the ad hoc retrieval task in TREC (Voorhees and Harman, 2001)

Due to the inherent vagueness and uncertainty of relevance, it has been argued that, instead of having a retrieval system to make a clear distinction between relevant and non-relevant documents, a TR system should rank the selected documents in the order of decreasing relevance, with the user deciding when to stop (Robertson, 1977). Thus modern retrieval systems generally assign a relevance value to each document and rank documents accordingly. The ranked list of documents is then presented to a user. Sometimes, after viewing the documents, the user may be able to tell the system which documents are relevant and which are not. Such *feedback* information can then be used by a system to improve the retrieval process, resulting in presumably better ranking results. This procedure is called *relevance feedback*.

To compute the relevance value of a document with respect to a query, a retrieval system must assume some retrieval model, which formally defines a relevance measure based on certain representation of a query and documents. Over the decades, many different types of retrieval models have been proposed and tested (Sparck Jones and Willett, 1997). A great diversity of approaches and methodology has developed, but no single unified retrieval model has proven to be most effective. The field has progressed in two different ways. On the one hand, theoretical studies of an underlying model have been developed; this direction is, for example, represented by the various kinds of logic models, e.g., (van Rijsbergen, 1986). On the other hand, there have been many empirical studies of models, including many variants of the vector space model, e.g., (Salton and Buckley, 1988). It has always been a significant scientific challenge to develop theoretically motivated models that also perform well empirically. A very interesting work in this direction is the development of the BM25 retrieval function, which was motivated by the 2-poisson probabilistic retrieval model and has proven to be quite effective in practice (Robertson and Walker, 1994; Robertson et al., 1995).

Recently, a new approach based on language modeling has been successfully applied to the problem of ad hoc retrieval (Ponte and Croft, 1998). This new approach is very promising, because of its foundations in statistical theory, the great deal of complementary work on language modeling in speech recognition and natural language processing, and the fact that very simple language modeling retrieval methods have performed quite well empirically. There is great potential of using statistical language models to develop new principled retrieval models.

In this thesis, we present a new text retrieval framework based on Bayesian decision theory that facilitates the development of new principled approaches to text retrieval using

statistical language models. In this framework, queries and documents are modeled using statistical language models (i.e., probabilistic models of text), user preferences are modeled through loss functions, and retrieval is cast as a risk minimization problem (a decision problem). The framework thus incorporates language models as natural components for retrieval, and provides a general connection between the problem of retrieval and that of statistical language modeling.

Through this framework, we address several limitations of traditional retrieval models. By choosing different loss functions and different query/document language models, we obtain different special cases of the risk minimization framework. We show how the risk minimization framework can unify several existing retrieval models, including the language modeling approach proposed recently, within one general probabilistic framework. By viewing the traditional models in this general framework we can see clearly what assumptions have been made in them, and thus understand better their limitations, enabling us to improve these models.

While traditional retrieval models rely heavily on ad hoc parameter tuning to achieve satisfactory retrieval performance, the use of language models makes it possible to exploit statistical estimation methods to improve retrieval performance and set retrieval parameters automatically. As a special case in the risk minimization framework, we present a two-stage language model that, according to extensive evaluation, achieves excellent retrieval performance without any ad hoc parameter tuning.

The use of language models in retrieval also provides more guidance on how to improve retrieval performance (e.g., through using more reasonable language models and/or more accurate estimation methods). As another special case of the risk minimization framework, we derive a Kullback-Leibler divergence retrieval model that can exploit feedback documents to improve the estimation of query models. Feedback has so far been dealt with heuristically in the language modeling approach to retrieval. The KL-divergence model provides a more natural way of performing feedback by treating it as query model updating. We propose two specific query model updating algorithms based on feedback documents. Evaluation indicates that both algorithms are effective for feedback, and outperform the baseline methods significantly.

The risk minimization retrieval framework further allows for incorporating user factors beyond the traditional notion of topical relevance. We present language models that can capture redundancy and sub-topics in documents, and study loss functions that can rank documents in terms of both relevance and sub-topic diversity. Evaluation shows that

the proposed language models can effectively capture redundancy and can outperform the relevance-based ranking method for the aspect retrieval task.

The thesis gives a new perspective on TR models. The risk minimization framework provides a general probabilistic retrieval framework that unifies several different existing TR models, including the recently proposed language modeling approach. The framework also provides a general connection between text retrieval and statistical language models, making it possible to apply statistical estimation methods to text retrieval. Furthermore, it can go beyond the traditional notion of topical relevance and potentially incorporate more user factors as retrieval criteria. The risk minimization framework opens up many new possibilities for developing principled approaches to text retrieval. It allows for exploring different retrieval models systematically through considering different loss functions and using different language models in the framework, as demonstrated by several special cases explored in this thesis.

Chapter 2

Review of Existing Retrieval Models

To identify relevant documents, a TR system must assume some specific measure of relevance between a document and a query. That is, an *operational* definition of a "relevant document" with respect to a query is needed. Thus, a fundamental problem in TR is to formalize the concept of relevance. A different formalization of relevance generally leads to a different TR model.

Over the decades, many different retrieval models have been proposed, studied, and tested. Their mathematical basis spans a large spectrum, including algebra, logic, probability and statistics. The existing models can be roughly grouped into three major categories, depending on how they define/measure relevance. In the first category, relevance is assumed to be correlated with the similarity between a query and a document. In the second category, a binary random variable is used to model relevance and probabilistic models are used to estimate the value of this relevance variable. In the third category, the relevance uncertainty is modeled by the uncertainty of inferring queries from documents or vice versa. We now discuss the three categories in details.

2.1 Similarity-based Models

In a similarity-based retrieval model, it is assumed that the relevance status of a document with respect to a query is correlated with the *similarity* between the query and the document at some level of representation; the more similar to a query a document is, the more relevant the document is assumed to be. In practice, we can use any similarity measure that preserves such a correlation to generate a relevance status value (RSV) for each document

and rank documents accordingly.

The vector space model is the most well-known model of this type (Salton et al., 1975a; Salton and McGill, 1983; Salton, 1989). In the vector space model, a document and a query are represented as two term vectors in a high-dimensional term space. Each term is assigned a weight that reflects its "importance" to the document or the query. Given a query, the relevance status value of a document is given by the similarity between the query vector and document vector as measured by some vector similarity measure, such as the cosine of the angle formed by the two vectors.

Formally, a document d may be represented by a document vector $\vec{d} = (x_1, x_2, ..., x_n)$, where n is the total number of terms and x_i is the weight assigned to term i. Similarly, a query q can be represented by a query vector $\vec{q} = (y_1, y_2, ..., y_n)$. The weight is usually computed based on the so-called TF-IDF weighting, which is a combination of three factors (Singhal, 2001): (1) the local frequency of the term (in a document or query); (2) the global frequency of the term in the whole collection; (3) document length. With the cosine measure, we have the following similarity function of the document and query:

$$sim(d, q) = \frac{\vec{d} \cdot \vec{q}}{\sqrt{\|\vec{d}\| \|\vec{q}\|}}$$

The vector space model naturally decomposes a retrieval model into three components: (1) a term vector representation of query; (2) a term vector representation of document; (3) a similarity/distance measure of the document vector and the query vector. However, the "synchronization" among the three components is generally unspecified; in particular, the similarity measure does not dictate the representation of a document or query. Thus, the vector space model is actually a general retrieval *framework*, in which the representation of query and documents as well as the similarity measure can all be arbitrary in principle.

Related to its generality, the vector space model can also be regarded as a procedural model of retrieval, in which the task of retrieval is naturally divided into two separate stages: indexing and search. The indexing stage explicitly has to do with representing the document and the query by the "indexing terms" extracted from the document and the query. The indexing terms are often assigned different weights to indicate their importance in describing a document or a query. The search stage has to do with evaluating the relevance value (i.e., the similarity) between a document vector and a query vector. The flexibility of the vector space model makes it easy to incorporate different indexing models. For example, the 2-Poisson probabilistic indexing model can be used to select indexing

terms and/or assign term weights (Harter, 1975; Bookstein and Swanson, 1975). Latent semantic indexing can be applied to reduce the dimension of the term space and to capture the semantic "closeness" among terms, and thus to improve the representation of the documents and query (Deerwester et al., 1990). A document can also be represented by a multinomial distribution over the terms, as in the distribution model of indexing proposed in (Wong and Yao, 1989).

In the vector space model, feedback is typically treated as query vector updating. A well-known approach is the Rocchio method, which simply adds the centroid vector of the relevant documents to the query vector and subtracts from it the centroid vector of the non-relevant documents with appropriate coefficients (Rocchio, 1971). In effect, this leads to an expansion of the original query vector, i.e., additional terms are extracted from the known relevant (and non-relevant) documents, and are added to the original query vector with appropriate weights (Salton and Buckley, 1990).

The extended Boolean (p-norm) model is a heuristic extension of the traditional Boolean model to perform document ranking, but it can also be regarded as a special case of the similarity model (Fox, 1983; Salton et al., 1983). The similarity function has a parameter p that controls the "strictness" of satisfying the constraint of a Boolean query, in such a way that it approaches a strict (conjunctive or disjunctive) Boolean model when p approaches infinity, but softens the conjunctive or disjunctive constraint and behaves more like a regular vector space similarity measure as p becomes smaller. However, the model must rely on some assumptions about the Boolean structure of a query, and has some undesirable mathematical properties (Rousseau, 1990). There has also been little, if any, large-scale evaluation of the model.

The vector space model is by far the most popular retrieval model due to its simplicity and effectiveness. The following is a typical effective weighting formula with pivoted document length normalization taken from (Singhal, 2001):

$$\sum_{t \in Q,D} \frac{1 + \ln(1 + \ln(tf))}{(1-s) + s\frac{dl}{avdl}} \times qtf \times \ln \frac{N+1}{df}$$

where s is an empirical parameter (usually 0.20), and

tf is the term's frequency in document
qtf is the term's frequency in query
N is the total number of documents in the collection
df is the number of documents that contain the term
dl is the document length, and
$avdl$ is the average document length.

The main criticism for the vector space model is that it provides no formal framework for the representation, making the study of representation inherently separate from the relevance estimation. The separation of the relevance function from the weighting of terms has the advantage of being flexible, but makes it very difficult to study the interaction of representation and relevance measurement. The semantics of a similarity/relevance function is highly dependent on the actual representation (i.e., term weights) of the query and the document. As a result, the study of representation in the vector space model has been so far largely heuristic. The two central problems in document and query representation are the extraction of indexing terms/units and the weighting of the indexing terms. The choice of different indexing units has been extensively studied, but no significant improvement has been achieved over the simplest word-based indexing (Lewis, 1992), though some more recent evaluation has shown more promising improvement on average through using linguistic phrases (Evans and Zhai, 1996; Strzalkowski, 1997; Zhai, 1997). Many heuristics have also been proposed to improve term weighting, but again, no weighting method has been found to be significantly better than the heuristic TF-IDF term weighting (Salton and Buckley, 1988). To address the variances in the length of documents, an effective weighting formula also needs to incorporate document length heuristically (Singhal et al., 1996). Salton et al. introduced the idea of the discrimination value of an indexing term (Salton et al., 1975b). The discrimination value of an indexing term is the increase or the decrease in the mean inter-document distance caused by adding the indexing term to the term space for text representation. They found that the middle frequency terms have a higher discrimination value. Given a similarity measure, the discrimination value provides a principled way of selecting terms for indexing. However, there are still two deficiencies. First, the discrimination value is not modeling relevance, but rather, relies on a given similarity measure. Second, it is only helpful for selecting indexing terms, but not for the weighting of terms.

The risk minimization framework we propose suggests a new formal similarity-based retrieval model in which the representation of query and documents are associated with statistical language models. The use of statistical language models makes it possible to

replace the traditional ad hoc tuning of parameters with the more principled estimation of parameters. The traditional vector space models can be regarded as special cases of this more general similarity model where the parameters are set heuristically. For example, we can represent a query and a document by a unigram language model, and thus easily derive a similarity model similar to the distribution model proposed in (Wong and Yao, 1989).

2.2 Probabilistic Relevance Models

In a probabilistic relevance model, we are interested in the question "What is the probability that *this* document is relevant to *this* query?" (Sparck Jones et al., 2000). Given a query, a document is assumed to be either relevant or non-relevant, but a system can never be sure about the true relevance status of a document, so it has to rely on a probabilistic relevance model to estimate it.

Formally, let random variables D and Q denote a document and query, respectively. Let R be a binary random variable that indicates whether D is relevant to Q or not. It takes two values which we denote as r ("relevant") and $\overline{r}$ ("not relevant"). The task is to estimate the probability of relevance, i.e., $p(R = r \mid D, Q)$. Depending on how this probability is estimated, there are several special cases of this general probabilistic relevance model.

First, $p(R = r \mid D, Q)$ can be estimated *directly* using a discriminative (regression) model. Essentially, the relevance variable R is assumed to be dependent on "features" that characterize the matching of D and Q. Such a regression model was probably first introduced with some success by Fox (Fox, 1983), where features such as term frequency, authorship, and co-citation were combined using linear regression. Fuhr and Buckley (Fuhr and Buckley, 1991) used polynomial regression to approximate relevance. Gey used logistic regression involving information such as query term frequency, document term frequency, IDF, and relative term frequency in the whole collection, and this model shows promising performance in three small testing collections (Gey, 1994). Regression models provide a principled way of exploring heuristic features and ideas. One important advantage of regression models is their ability to learn from all the past relevance judgments, in the sense that the parameters of a model can be estimated based on all the relevance judgments, including the judgments for *different* queries or documents. However, because regression models are based on heuristic features in the first place, lots of empirical experimentation would be needed in order to find a set of good features. A regression model thus provides only limited guidance for extending a retrieval model.

Alternatively, $p(R = r \mid D, Q)$ can be estimated *indirectly* using a generative model in the following way (Lafferty and Zhai, 2003):

$$p(R = r \mid D, Q) \;=\; \frac{p(D, Q \mid R = r)\, p(R = r)}{p(D, Q)}\,.$$

Equivalently, we may use the following log-odds ratio to rank documents:

$$\log \frac{p(r \mid D, Q)}{p(\overline{r} \mid D, Q)} \;=\; \log \frac{p(D, Q \mid r)\, p(r)}{p(D, Q \mid \overline{r})\, p(\overline{r})}\,.$$

There are two different ways to factor the conditional probability $p(D, Q \mid R)$, corresponding to "document generation" and "query generation."

With document generation, $p(D, Q \mid R) = p(D \mid Q, R)p(Q \mid R)$, so we end up with the following ranking formula:

$$\log \frac{p(r \mid D, Q)}{p(\overline{r} \mid D, Q)} = \log \frac{p(D \mid Q, r)}{p(D \mid Q, \overline{r})} + \log \frac{p(r \mid Q)}{p(\overline{r} \mid Q)}$$

Essentially, the retrieval problem is formulated as a two-category document classification problem, though we are only interested in ranking the classification likelihood, rather than actually assigning class labels. Operationally, two models are estimated for each query, one modeling relevant documents, the other modeling non-relevant documents. Documents are then ranked according to the posterior probability of relevance.

Most classic probabilistic retrieval models (Robertson and Sparck Jones, 1976; van Rijsbergen, 1979; Fuhr, 1992) are based on document generation. The Binary Independence Retrieval (BIR) model (Robertson and Sparck Jones, 1976; Fuhr, 1992) is perhaps the most well known classical probabilistic model. The BIR model assumes that terms are independently distributed in each of the two relevance models, so it essentially uses the Naïve Bayes classifier for document ranking (Lewis, 1998). [1] The BIR retrieval formula is the following (Robertson and Sparck Jones, 1976; Lafferty and Zhai, 2003):

$$\log \frac{p(r \mid D, Q)}{p(\overline{r} \mid D, Q)} \stackrel{\text{rank}}{=} \sum_{t \in D \cap t \in Q} \log \frac{p(t \mid Q, r)(1 - p(t \mid Q, \overline{r}))}{(1 - p(t \mid Q, r))p(t \mid Q, \overline{r})}$$

where $\stackrel{\text{rank}}{=}$ means equivalent in terms of being used for ranking documents.

There have been several efforts to improve the binary representation. van Rijsbergen extended the binary independence model by capturing some term dependency as defined by

[1] The required underlying independence assumption for the final retrieval formula is actually weaker (Cooper, 1991).

a minimum-spanning tree weighted by average mutual information (van Rijbergen, 1977). The dependency model achieved significant increases in retrieval performance over the independence model. However, the evaluation was only done on very small collections, and the estimation of many more parameters is a problem in practice (Harper and van Rijsbergen, 1978). Croft investigated how the heuristic term significance weight can be incorporated into probabilistic models in a principled way (Croft, 1981). Another effort to improve document representation involves introducing the term frequency directly into the model by using a multiple 2-Poisson mixture representation of documents (Robertson et al., 1981). This model has not shown empirical improvement in retrieval performance directly, but an approximation of the model using a simple TF formula turns out to be quite effective (Robertson and Walker, 1994). The heuristic retrieval formula BM25 has been successfully used in City University's Okapi system and several other TREC systems (Voorhees and Harman, 2001). A different way of introducing the term frequency into the model, not directly proposed but implied by a lot of work in text categorization, is to regard a document as being generated from a unigram language model (Kalt, 1996; McCallum and Nigam, 1998).

With query generation, $p(D, Q \mid R) = p(Q \mid D, R)p(D \mid R)$, so we end up with the following ranking formula:

$$\log \frac{p(r \mid D, Q)}{p(\overline{r} \mid D, Q)} = \log \frac{p(Q \mid D, r)}{p(Q \mid D, \overline{r})} + \log \frac{p(r \mid D)}{p(\overline{r} \mid D)}$$

Under the assumption that conditioned on the event $R = \overline{r}$, the document D is independent of the query Q, i.e., $p(D, Q \mid R = \overline{r}) = p(D \mid R = \overline{r})\, p(Q \mid R = \overline{r})$, the formula becomes

$$\log \frac{p(r \mid D, Q)}{p(\overline{r} \mid D, Q)} \stackrel{\text{rank}}{=} \log p(Q \mid D, r) + \log \frac{p(r \mid D)}{p(\overline{r} \mid D)}$$

There are two components in this model. The major component $p(Q \mid D, r)$ can be interpreted as a "relevant query model" conditioned on a document. That is, $p(Q \mid D, r)$ is the probability that a user who likes document D would use Q as a query. The second component $p(r \mid D)$ is a prior that can be used to encode a user's bias on documents.

Models based on query generation have been explored in (Maron and Kuhns, 1960), (Fuhr, 1992) and (Lafferty and Zhai, 2001b). The probabilistic indexing model proposed in (Maron and Kuhns, 1960) is the first probabilistic retrieval model, in which the indexing terms assigned to a document are weighted by the probability that a user who likes the

document would use the term in the query. That is, the weight of term t for document D is $p(t \mid D, r)$. However, the estimation of the model is based on the user's feedback, not the content of D. The Binary Independence Indexing (BII) model proposed in (Fuhr, 1992) is another special case of the query generation model. It allows the description of a document (with weighted terms) to be estimated based on arbitrary queries, but the specific parameterization makes it hard to estimate all the parameters in practice. In (Lafferty and Zhai, 2001b), it has been shown that the recently proposed language modeling approach to retrieval is also a special probabilistic relevance model when query generation is used to decompose the generative model. This work provides a relevance-based justification for this new family of probabilistic models based on statistical language modeling.

The language modeling approach was first introduced in (Ponte and Croft, 1998) and later explored in (Hiemstra and Kraaij, 1998; Miller et al., 1999; Berger and Lafferty, 1999; Song and Croft, 1999), among others. The estimation of a language model based on a document (i.e., the estimation of $p(. \mid D, r)$) is the key component in the language modeling approach. Indeed, most work in this direction differs mainly in the language model used and the method of language model estimation. Smoothing document language models with some kind of collection language model has been very popular in the existing work. For example, geometric smoothing was used in (Ponte and Croft, 1998); linear interpolation smoothing was used in (Hiemstra and Kraaij, 1998; Berger and Lafferty, 1999), and was viewed as a 2-state hidden Markov model in (Miller et al., 1999). Berger and Lafferty explored "semantic smoothing" by estimating a "translation model" for mapping a document term to a query term, and reported significant improvements over the baseline language modeling approach through the use of translation models (Berger and Lafferty, 1999).

The language modeling approach has two important contributions. First, it introduces a new effective probabilistic ranking function based on query generation. While the earlier query generation models have all found estimating the parameters difficult, the model proposed in (Ponte and Croft, 1998) explicitly addresses the estimation problem through the use of statistical language models. Second, it reveals the connection between the difficult problem of text representation in IR and the language modeling techniques that have been well-studied in other application areas such as statistical machine translation and speech recognition, making it possible to exploit various kinds of language modeling techniques to address the representation problem.[2]

[2]The use of a multinomial model for documents was actually first introduced in (Wong and Yao, 1989), but was not exploited as a language model.

While based on the same notion of relevance and probabilistically equivalent, the classic document generation probabilistic models and the language modeling approach have several important differences from an estimation perspective, as they involve different parameters for estimation. When no relevance judgments are available, it is easier to estimate $p(Q \mid D, r)$ in the language modeling approach than to estimate $p(D \mid Q, r)$ in the classic probabilistic models. Intuitively, it is easier to estimate a model for "relevant queries" based on a document than to estimate a model for relevant documents based on a query. Indeed, the BIR model has encountered difficulties in estimating $p(t \mid Q, r)$ and $p(t \mid Q, \overline{r})$ when no explicit relevance information is available. Typically, $p(t \mid Q, r)$ is set to a constant and $p(t \mid Q, \overline{r})$ is estimated under the assumption that the whole collection of documents is non-relevant (Croft and Harper, 1979; Robertson and Walker, 1997). Recently, Lavrenko and Croft made progress in estimating the relevance model without relevance judgments by exploiting language modeling techniques (Lavrenko and Croft, 2001). When explicit relevance judgments are available, the classic models, being based on document generation, have the advantage of being able to improve the estimation of the component probabilistic models naturally by exploiting such explicit relevance information. This is because the relevance judgments from a user provide direct training data for estimating $p(t \mid Q, r)$ and $p(t \mid Q, \overline{r})$, which can then be applied to *new* documents. The same relevance judgments can also provide direct training data for improving the estimate of $p(t \mid D, r)$ in the language modeling approach, but only for those judged relevant documents. Thus, the directly improved models can *not* be expected to improve our ranking of other un-judged documents. Interestingly, such improved models can potentially be beneficial for new queries – a feature unavailable in document generation models.

Instead of imposing a strict document generation or query generation decomposition of $p(D, Q \mid R)$, one can also "generate" a document-query pair simultaneously. Mittendorf & Schauble explored a passage-based generative model using the Hidden Markov Model (HMM), which can be regarded as such a case (Mittendorf and Schauble, 1994). In this work, a document query pair is represented as a sequence of symbols, each corresponding to a term in a particular position of the document. All term tokens are clustered in terms of the similarity between the token and the query. In this way, a term token in a particular position of a document can be mapped to a symbol that represents the cluster the token belongs to. Such symbol sequences are modeled as the output from an HMM with two states, one corresponding to relevant passage and the other the background noise. The relevance value is then computed based on the likelihood ratio of the sequence given the passage HMM model and the background model.

Empirically, probabilistic relevance models have shown good performance. Indeed, a simple approximation of the 2-Poisson probabilistic model, which has led to the BM25 retrieval formula used in the Okapi system, has been very effective (Robertson and Walker, 1994; Robertson et al., 1995). The language modeling approaches have also been shown to perform very well (Ponte and Croft, 1998; Hiemstra and Kraaij, 1998; Miller et al., 1999). The BM25 formula is shown below, following the notations used in (Singhal, 2001):

$$\sum_{t \in Q,D} \ln \frac{N - df + 0.5}{df + 0.5} \times \frac{(k_1 + 1)tf}{(k_1(1-b) + b\frac{dl}{avdl}) + tf} \times \frac{(k_3 + 1)qtf}{k_3 + qtf}$$

where, $k_1 \in [1.0, 2.0]$, b (usually 0.75), and $k_3 \in [0, 1000]$ are parameters, and other variables have the same meaning as in the vector space retrieval formula described in the previous section.

Probabilistic relevance models can be shown to be a special case of the risk minimization framework when a "constant-cost" loss function is used. In the risk minimization framework, we also maintain a separate generative model for queries and documents respectively. With a slightly different loss function, we derive a two-stage language modeling approach to text retrieval and propose methods for automatically estimating the parameters for a two-stage smoothing strategy. According to extensive evaluation, the two-stage smoothing method gives excellent retrieval performance through completely automatic parameter settings.

2.3 Probabilistic Inference Models

In a probabilistic inference model, the relevance uncertainty of a document with respect to a query is modeled by the uncertainty associated with inferring/proving the query from the document. Depending on how one defines what it means by "proving a query from a document," different inference models are possible.

van Rijsbergen introduced a logic-based probabilistic inference model for text retrieval (van Rijsbergen, 1986), in which, a document is relevant to a query if and only if the query can be proved from the document. The Boolean retrieval model can be regarded as a simple case of this model. To cope with the inherent uncertainty in relevance, van Rijsbergen introduced a logic for probabilistic inference, in which the probability of a conditional, such as $p \rightarrow q$, can be estimated based on the notion of possible worlds. In (Wong and Yao, 1995), Wong and Yao extended the probabilistic inference model and developed a general

probabilistic inference model which subsumes several other TR models such as Boolean, vector space, and the classic probabilistic models. Fuhr shows that some particular form of the language modeling approach can also be derived as a special case of the general probabilistic inference model (Fuhr, 2001).

While theoretically interesting, the probabilistic inference models all must rely on further assumptions about the representation of documents and queries in order to obtain an operational retrieval formula. The choice of such representations is in a way outside the model, so there is little guidance on how to choose or how to improve a representation.

The inference network model is also based on probabilistic inference (Turtle and Croft, 1991). It is essentially a Bayesian belief network that models the dependency between the satisfaction of a query and the observation of documents. The estimation of relevance is based on the computation of the conditional probability that the query is satisfied given that the document is observed. Other similar uses of the Bayesian belief network in retrieval are presented in (Fung and Favero, 1995; Ribeiro and Muntz, 1996; Ribeiro-Neto et al., 2000). The inference network model is a much more general formalism than most of the models that we have discussed above. With different ways to realize the probabilistic relationship between the observation of documents and the satisfaction of the user's information need, one can obtain many different existing specific TR models, such as Boolean, extended Boolean, vector space, and conventional probabilistic models. More importantly, the inference network model can potentially go beyond the traditional notion of topical relevance; indeed, the goal of inference is a very general one, and at its highest level, the framework is so general that it can accommodate almost any probabilistic model. The generality makes it possible to combine multiple evidence, including different formulations of the same query. The query language based directly on the model has been an important and practical contribution to IR technology.

However, despite its generality, the inference network framework says little about how one can further decompose the general probabilistic model. As a result, operationally, one usually has to set probabilities based on heuristics, as was done in the Inquery system (Callan et al., 1992).

Kowk's network model may also be considered as performing a probabilistic inference (Kwok, 1995), though it is based on spread activation.

In general, the probabilistic inference models address the issue of relevance in a very general way. In some sense, the lack of a commitment to specific assumptions in these general models has helped to maintain their generality as retrieval models. But this also

deprives them of "predictive power" as a theory. As a result, they generally provide little guidance on how to refine the general notion of relevance.

The risk minimization framework is also quite general. Indeed, we will be able to show that many existing models are special cases of risk minimization. Furthermore, the framework goes beyond the traditional notion of topical relevance, just like the inference network framework, and it allows for incorporating multiple user factors as retrieval criteria. However, the risk minimization framework is different from the probabilistic inference models and other Bayesian belief network models in that it provides an explicit and direct connection to (query and document) language models. Techniques of language modeling can thus be brought into an operational retrieval model easily. In this sense, it is a much more refined and operational framework than probabilistic inference models. The thesis will explore several such possibilities.

2.4 Summary

A large number of different retrieval approaches have been proposed and studied, and a tremendous amount of effort has been devoted to the evaluation of various kinds of approaches, especially in the context of TREC evaluation (Voorhees and Harman, 2001). There has been a lot of progress in both developing a retrieval theory and improving empirical performance. However, the existing research has several limitations.

First, the integration of theory and practice in text retrieval has so far been quite weak. Theoretical guidance and formal principles have rarely led to good performance directly; a lot of heuristic parameter tuning must be used in order to achieve good performance. Parameter tuning is generally difficult due to the fact that the optimal setting of parameters is often collection/query dependent and the parameters may interact with each other in a complicated way. This can be seen from the following excerpt about parameter tuning in the well-known Okapi system – one of the best performing TR systems today (The weighting formula is described in Section 2.2):

> "k_1,b and k_3 are parameters which depend on the nature of the queries and possibly on the database; k_1 and b default to 1.2 and 0.75 respectively, but smaller values of b are sometimes advantageous; in long queries k_3 is often set to 7 or 1000 (effectively infinite)" (Robertson and Walker, 1999).

It is thus a significant scientific challenge to develop principled retrieval approaches that

also perform well empirically.

In this thesis, we present a new text retrieval framework that facilitates the development of new principled approaches to text retrieval through the use of statistical language models. Statistical language models provide a principled way to model text documents and queries, making it possible to set retrieval parameters through statistical estimation methods. In particular, as a special case of the risk minimization framework, a two-stage language model is shown to achieve excellent retrieval performance without any ad hoc parameter tuning.

The existing work on the language modeling approach, while promising, has encountered some difficulty in incorporating a feedback mechanism, which is an important component in a retrieval system. Specifically, feedback has generally been dealt with in an ad hoc way. Typically, a set of terms are selected from the relevant documents (or documents assumed to be relevant in the situation of blind/pseudo feedback), and the query is augmented by the set of terms with some weight (Ponte, 1998; Miller et al., 1999; Ng, 2000). However, this creates a conceptual inconsistency for the query, since the original query is normally a piece of text, while the expanded query would be text plus some weighted terms. The difficulty is largely due to the lack of a query model in the existing work.

As another instance of the risk minimization framework, we derive a KL-divergence retrieval model, which covers the language modeling approach as a special case. The KL-divergence model explicitly deals with the query model and can perform feedback more naturally by treating it as query model updating.

Another fundamental limitation of most traditional retrieval models is the strong assumptions made about relevance (e.g., the independent relevance assumption and topical relevance assumption); these assumptions do not usually hold in reality. In real applications, the retrieval criteria may involve other factors such as redundancy. Thus most traditional models are inadequate for modeling more realistic retrieval criteria.

In the risk minimization framework, the retrieval problem is formalized as a statistical decision problem in a very general way. The generality not only allows it to unify many different traditional retrieval models within one single general probabilistic framework, but also makes it possible to go beyond the traditional notion of independent topical relevance and capture more realistic retrieval criteria. We explore language models that can capture the redundancy and sub-topics in documents, and study the loss functions that can rank documents based on both relevance and sub-topic diversity.

The risk minimization framework opens up many new possibilities for developing principled approaches to text retrieval, and makes it possible to explore different retrieval mod-

els more systematically, through considering different loss functions and using different language models in the framework. The thesis explores several such special cases.

Chapter 3

A Risk Minimization Framework for Text Retrieval

In this chapter, we present the risk minimization framework. Since the risk minimization formulation is based on Bayesian decision theory and statistical language modeling, we first give a brief introduction to both.

3.1 Bayesian Decision Theory

Bayesian decision theory provides a solid theoretical foundation for thinking about problems of action and inference under uncertainty (Berger, 1985). The basic idea can be explained by considering the following formulation of a decision problem.

We assume that an observation $\mathbf{x}$ is made on a random variable X whose distribution depends on the parameter θ, which can be interpreted as representing the true state of the nature. Let $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ be all the possible actions about θ. We further assume that a loss function $L(a_i, \theta)$ is associated with each action and each value of θ which specifies our decision preferences. The task is to make a decision on which action to take.

In order to evaluate each action, we consider the *Bayesian expected loss* (or *risk*) associated with taking action a_i given that $\mathbf{x}$ is observed:

$$R(a_i \mid \mathbf{x}) = \int_{\theta} L(a_i, \theta) p(\theta \mid \mathbf{x}) d\theta$$

Bayesian decision theory states that given $\mathbf{x}$, the optimal decision is to choose a *Bayes action*, i.e., an action $\mathbf{a}^*$ that minimizes the conditional expected risk (Berger, 1985):

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} R(\mathbf{a} \mid \mathbf{x})$$

3.2 Statistical Language Models

A statistical language model is a probabilistic mechanism for "generating" a sequence of words or text (Jelinek, 1997; Rosenfeld, 2000). It thus defines a distribution over all the possible word sequences. The simplest language model is the unigram language model, which is essentially a word distribution. With a unigram language model, the text is generated through generating each word independently. Thus, for example, the probability of generating a piece of text $w_1 w_2 ... w_n$ according to a unigram language model $p(. \mid \theta)$ would be

$$p(w_1 w_2 ... w_n \mid \theta) = \prod_{i=1}^{n} p(w_i \mid \theta)$$

More complex language models may capture word orders or even the structure of the text (Jelinek, 1997; Rosenfeld, 2000). For example, trigram language models would generate a word based on the two previously generated words, thus capturing the local ordering of words. Maximum entropy language models can capture long-distance dependency among words. Probabilistic context-free grammars are structure-based generative models of text, allowing for the incorporation of structural constraints. In general, statistical language models provide a principled way to capture quantitatively the uncertainty associated with the use of natural language, and they have found many applications in a variety of language technology tasks, especially speech recognition. Just recently have they been applied to text retrieval.

Although we ultimately need to explore more sophisticated models, in this thesis, we only study the simplest unigram language model for the following reasons: (1) Text retrieval generally involves a large amount of text information, which puts constraints on the efficiency of any algorithm to be evaluated. More sophisticated language models would significantly increase the computational complexity for retrieval, making large-scale evaluation practically infeasible. (2) In many cases, e.g., when estimating a document language model or a query language model, we are working on an extremely limited amount of data, so the estimation of complex models may not be reliable. In other words, the sparseness of data puts a constraint on the complexity of the model that we can estimate accurately. (3) Even with the simplest language models, existing work has been able to demonstrate very

promising retrieval performance. (4) Once we have a good understanding of the simple models, we will understand better what kind of sophisticated models may be expected to further improve the performance.

3.3 Retrieval as a Decision Problem

In general, a retrieval system can be regarded as an interactive information service system that answers a user's query by presenting a list of documents. After seeing the presented documents, the user may reformulate a query, which is then executed by the system to produce another list of documents to present. The cycle continues like this.

At each cycle, the retrieval system needs to choose a subset of documents and present them to the user in some way, based on the information available to the system, which includes the current user, the user's query, the sources of documents, and a specific document collection. In terms of Bayesian decision theory, our observations are the user $\mathcal{U}$, the query $\mathbf{q}$, the document sources $\vec{\mathcal{S}} = (\mathcal{S}_1, ..., \mathcal{S}_N)$, and the collection of documents $\mathcal{C}$. Some of these will be regarded as observations on some random variables. Specifically, we view a query as being the output of some probabilistic process associated with the user $\mathcal{U}$, and similarly, we view a document as being the output of some probabilistic process associated with an author or document source $\mathcal{S}_i$. A query (document) is the result of choosing a model, and then generating the query (document) using that model. A set of documents is the result of generating each document independently, possibly from a different model. (The independence assumption is not essential, and is made here only to simplify the presentation.) The query model could, in principle, encode detailed knowledge about a user's information need and the context in which they make their query. Similarly, the document model could encode complex information about a document and its source or author.

More formally, let θ_Q denote the parameters of a query model, and let θ_D denote the parameters of a document model. A user $\mathcal{U}$ generates a query by first selecting θ_Q, according to a distribution $p(\theta_Q \,|\, \mathcal{U})$. Using this model, a query $\mathbf{q}$ is then generated with probability $p(\mathbf{q} \,|\, \theta_Q)$. Note that since a user can potentially use the same text query to mean different information needs, strictly speaking, the variable $\mathcal{U}$ should be regarded as corresponding to a user with the *current* context. Since this does not affect the presentation of the framework, we will simply refer to $\mathcal{U}$ as a user. Similarly, the source selects a document model θ_D according to a distribution $p(\theta_D \,|\, \mathcal{S})$, and then uses this model to generate a document $\mathbf{d}$ according to $p(\mathbf{d} \,|\, \theta_D)$. Thus, we have Markov chains $\mathcal{U} \rightarrow \theta_Q \rightarrow \mathbf{q}$ and $\mathcal{S} \rightarrow \theta_D \rightarrow \mathbf{d}$.

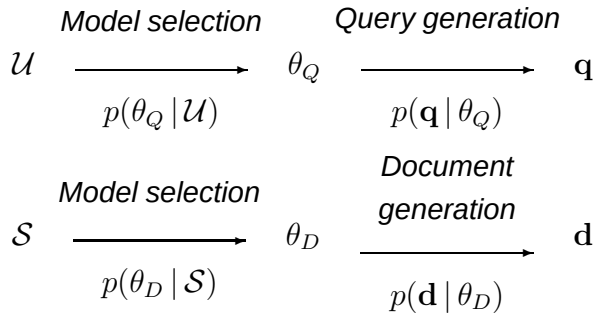

Figure 3.1: Generative model of query $\mathbf{q}$ and document $\mathbf{d}$.

This is illustrated in Figure 3.1

Let $\mathcal{C} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_N\}$ be a collection of documents obtained from sources $\vec{\mathcal{S}} = (\mathcal{S}_1, ..., \mathcal{S}_N)$. We denote by θ_i the model that generates document $\mathbf{d}_i$. Our observations are thus $\mathcal{U}$, $\mathbf{q}$, $\vec{\mathcal{S}}$, and $\mathcal{C}$.

Now, what about the actions? An action corresponds to a possible response of the system to a query. For example, one can imagine that the system would return an unordered subset of documents to the user. Alternatively, a system may decide a ranking of documents and present a ranked list of documents. Yet another possibility is to cluster the (relevant) documents and present a structured view of documents.

Generally, we can think of a retrieval action as a *compound decision* involving *selecting* a subset of documents D from $\mathcal{C}$ and *presenting* them to the user who has issued query $\mathbf{q}$ according to some presentation strategy π. Let Π be the set of all possible presentation strategies. We can represent all actions by $\mathcal{A} = \{(D_i, \pi_i)\}$, where $D_i \subseteq \mathcal{C}$ is a subset of $\mathcal{C}$ (results) and $\pi_i \in \Pi$ is some presentation strategy.

In the general framework of Bayesian decision theory, to each such action $a_i = (D_i, \pi_i) \in \mathcal{A}$ there is associated a *loss* $L(a_i, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))$, which in general depends upon all of the parameters of our model $\theta \equiv (\theta_Q, \{\theta_i\}_{i=1}^N)$ as well as any relevant user factors $F(\mathcal{U})$ and document source factors $F(\vec{\mathcal{S}})$.

In this framework, the *expected risk of action* a_i is given by

$$R(D_i, \pi_i \,|\, \mathcal{U}, \mathbf{q}, \vec{\mathcal{S}}, \mathcal{C}) \;\; = \;\; \int_\Theta L(D_i, \pi_i, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))\, p(\theta \,|\, \mathcal{U}, \mathbf{q}, \vec{\mathcal{S}}, \mathcal{C})\, d\theta$$

where the posterior distribution is given by

$$p(\theta \,|\, \mathcal{U}, \mathbf{q}, \vec{\mathcal{S}}, \mathcal{C}) \;\; \propto \;\; p(\theta_Q \,|\, \mathbf{q}, \mathcal{U}) \prod_{i=1}^{N} p(\theta_i \,|\, \mathbf{d}_i, \vec{\mathcal{S}})$$

The Bayes decision rule is then to choose the action $\mathbf{a}^*$ with the least expected risk:

$$\mathbf{a}^* \;=\; (D^*, \pi^*) \;=\; \arg\min_{D,\pi} R(D, \pi \,|\, \mathcal{U}, \mathbf{q}, \vec{\mathcal{S}}, \mathcal{C})$$

That is, to select D^* and present D^* with strategy π^*.

Note that this gives us a very general formulation of retrieval as a decision problem, which involves searching for D^* and π^* simultaneously. The presentation strategy can be fairly arbitrary in principle, e.g., presenting documents in a certain order, presenting a summary of the documents, or presenting a clustering view of the documents. Practically, however, we need to be able to quantify the loss associated with a presentation strategy.

In the following sections, we consider several special cases.

3.4 Set-based Retrieval

Let us consider the case when the loss function does *not* depend on the presentation strategy, which means that all we care about is to select an optimal subset of documents for presentation. In this case, the risk minimization framework leads to the following general set-based retrieval method.

$$\begin{aligned} D^* &= \arg\min_{D} R(D \,|\, \mathcal{U}, \mathbf{q}, \vec{\mathcal{S}}, \mathcal{C}) \\ &= \arg\min_{D} \int_{\Theta} L(D, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))\, p(\theta \,|\, \mathcal{U}, \mathbf{q}, \vec{\mathcal{S}}, \mathcal{C})\, d\theta \end{aligned}$$

The loss function can encode the user's preferences on the selected subset. Generally, the loss function will have to do with the *relevance* status of the documents selected so that the optimal subset should contain the documents that are most likely relevant. But other preferences, such as the desired diversity and the desired size of a subset, can also be captured by an appropriate loss function.

The traditional Boolean retrieval model can be viewed as a special case of this general set-based retrieval framework, where we have no uncertainty about the query models and document models (e.g., $\theta_Q = \mathbf{q}$ and $\theta_i = \mathbf{d}_i$), and the following loss function is used:

$$L(D, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}})) = \sum_{\mathbf{d} \in D} -\delta(\mathbf{d}, \mathbf{q})$$

where $\delta(\mathbf{d}, \mathbf{q}) = 1$ if and only if document $\mathbf{d}$ satisfies the Boolean query $\mathbf{q}$; otherwise $\delta(\mathbf{d}, \mathbf{q}) = -1$. This loss function is actually quite general, in the sense that if we allow $\delta(\mathbf{d}, \mathbf{q})$ to be any *deterministic* retrieval rule applied to query $\mathbf{q}$ and document $\mathbf{d}$, such that $\delta(\mathbf{d}, \mathbf{q}) > 0$ if $\mathbf{d}$ is relevant to $\mathbf{q}$, otherwise $\delta(\mathbf{d}, \mathbf{q}) < 0$, then the loss function would always result in a retrieval strategy that involves making an *independent* binary retrieval decision for each document according to δ. In particular, the function δ can be defined on a structured query. There are many other ways to specialize the set-based retrieval method, but exploring them is not the main focus of this thesis.

A major disadvantage of set-based retrieval is that it may be hard for the user to control the number of results. Often there is either a zero-return (when the query is too constrained) or an overwhelming number of results (when the query is under-constrained). It is also hard for the user to locate the most relevant ones without reading through many likely non-relevant documents. Thus, it may be more useful to rank documents, instead of making a "hard" retrieval decision on each document.

3.5 Rank-based Retrieval

Let us now consider a different special case of the risk minimization framework, where the selected documents are presented to the user as a ranked list of documents, so a possible presentation strategy corresponds to a possible *ranking* of documents.

Such a ranking strategy has been assumed in most modern retrieval systems and models. Formally, we may denote an action by $a_i = (D_i, \pi_i)$, where π_i is a complete ordering on D_i. [1] Taking action a_i would then mean to present the selected documents in D one by one in the order given by π_i. This means that we can denote an action by a *sequence* of documents. So, we will write $a_i = (d_{\pi_i^1}, d_{\pi_i^2}, ..., d_{\pi_i^k})$, where π_i^j is the index of the document ranked at the j-th rank according to the permutation mapping π_i.

Let us further assume that our actions essentially involve different rankings of documents in the *whole* collection $\mathcal{C}$. That is, $\mathcal{A} = \{(\mathcal{C}, \pi_i)\}$, where π_i is a permutation over $[1..N]$, i.e., a complete ordering of the N documents in $\mathcal{C}$. To simplify our notations, we will use π_i to denote action $a_i = (\mathcal{C}, \pi_i)$.

[1] We could allow partial ordering in principle, but here we only consider complete ordering.

In this case, the optimal Bayes decision is given by the following general *ranking* rule:

$$\begin{aligned}\pi^* &= \arg\min_{\pi} R(\pi|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) \\ &= \arg\min_{\pi} \int_{\Theta} L(\pi, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))p(\theta|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}})d\theta\end{aligned}$$

where $\theta = (\theta_Q, \{\theta_i\}_{i=1}^N)$.

We see that the loss function is now discriminating different possible rankings of documents.

How do we characterize the loss associated with a *ranking* of documents? Presenting documents by ranking implies that the user would apply some stopping criterion – the user would read the documents in order and stop wherever is appropriate. Thus, the *actual* loss (or equivalently utility) of a ranking would depend on where the user actually stops. That is, the utility is affected by the user's browsing behavior, which we could model through a probability distribution over all the ranks at which a user might stop. Given this setup, we can now define the loss for a ranking as the *expected* loss under the assumed "stopping distribution."

Formally, let s_i denote the probability that the user would stop reading after seeing the top i documents. We have $\sum_{i=1}^N s_i = 1$. We can treat $s_1, ..., s_N$ as user factors given by $F(\mathcal{U})$.

$$L(\pi, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}})) = \sum_{i=1}^{N} s_i l(\pi(1:i), \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))$$

where, $l(\pi(1:i), \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))$ is the actual loss that would be incurred if the user actually views the first i documents according to π. Note that $L(\pi, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))$ and $l(\pi, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))$ are different: the former is the *expected* loss of the ranking under the user's "stopping probability distribution," while the latter is the *exact* loss of the ranking when the user actually views the whole list.

Assuming that the user would view the documents in the order of presented, and the total loss of viewing i documents is the sum of the loss associated with viewing each individual document, we have the following reasonable decomposition of the loss:

$$l(\pi(1:i), \theta, F(\mathcal{U}), F(\vec{\mathcal{S}})) = \sum_{j=1}^{i} l(d_{\pi^j}|d_{\pi^1}, ..., d_{\pi^{j-1}}, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}}))$$

where $l(d_{\pi^j}|d_{\pi^1},...,d_{\pi^{j-1}},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))$ is the *conditional* loss of viewing d_{π^j} given that the user has already viewed $(d_{\pi^1},...,d_{\pi^{j-1}})$.

Putting all these together, we have

$$\begin{aligned}\pi^* &= \arg\min_{\pi} R(\pi|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \\ &= \arg\min_{\pi} \sum_{i=1}^{N} s_i \sum_{j=1}^{i} \int_{\Theta} l(d_{\pi^j}|d_{\pi^1},...,d_{\pi^{j-1}},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))p(\theta|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}})d\theta\end{aligned}$$

Define the following *conditional* risk

$$r(\mathbf{d}_k|\mathbf{d}_1,...,\mathbf{d}_{k-1},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \stackrel{\text{def}}{=} \int_{\Theta} l(\mathbf{d}_k|\mathbf{d}_1,...,\mathbf{d}_{k-1},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))p(\theta|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}})d\theta$$

which can be interpreted as the expected risk of the user's viewing document $\mathbf{d}_k$ given that $\mathbf{d}_1,...,\mathbf{d}_{k-1}$ have already been previously viewed. We can write

$$\begin{aligned}R(\pi|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) &= \sum_{i=1}^{N} s_i \sum_{j=1}^{i} r(d_{\pi^j}|d_{\pi^1},...,d_{\pi^{j-1}},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \\ &= \sum_{j=1}^{N} (\sum_{i=j}^{N} s_i) r(d_{\pi^j}|d_{\pi^1},...,d_{\pi^{j-1}},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}})\end{aligned}$$

This is the general framework for *ranking* documents within the risk minimization framework. It basically says that the optimal ranking minimizes the expected conditional loss (under the stopping distribution) associated with sequentially viewing each document.

We see that the optimal ranking depends on the stopping distribution s_i. If a user tends to stop early, the optimal decision would be more affected by the loss associated with the top ranked documents; otherwise, it would be somehow "equally" affected by the loss associated with all the documents. Thus, the stopping probability distribution provides a way to model a "high-precision" (early stopping) preference or a "high-recall" (late stopping) preference. The sequential decomposition of the loss is reasonable when presenting a ranked list to the user. Clearly, when using other presentation strategies (e.g., clustering), such decomposition would not be appropriate.

We now consider two different cases of the loss function.

3.5.1 Independent Loss

Let us first consider the case where the loss of viewing each document is *independent* of that of viewing others. That is,

$$l(d_{\pi^j}|d_{\pi^1},...,d_{\pi^{j-1}},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}})) = l(d_{\pi^j},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))$$

which means

$$l(\pi(1:i),\theta,F(\mathcal{U}),F(\vec{\mathcal{S}})) = \sum_{j=1}^{i} l(d_{\pi^j},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))$$

In this case, the expected risk for ranking π is

$$\begin{aligned} R(\pi|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) &= \sum_{i=1}^{N} s_i \sum_{j=1}^{i} r(d_{\pi^j}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \\ &= \sum_{j=1}^{N} (\sum_{i=j}^{N} s_i) r(d_{\pi^j}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \end{aligned}$$

We see that the risk of π is a weighted sum of the risk of viewing each individual document. As the rank increases, the weight decreases, with the weight on the first rank being the largest (i.e., $\sum_{i=1}^{N} s_i$). Thus, the optimal ranking π^*, *independent* of $\{s_i\}$, is in ascending order of the individual risk:

$$r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) = \int_{\Theta} l(\mathbf{d},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))p(\theta|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}})d\theta \tag{3.1}$$

This is equivalent to the situation when we assume each possible action is to present a *single* document. The loss function $l(\mathbf{d},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))$ can be interpreted as the loss associated with presenting/viewing document $\mathbf{d}$. A complete specification of the loss function will generally force us to make explicit the assumed ranking criterion and notion of relevance. We now show that this risk function covers several existing retrieval models as special cases (Lafferty and Zhai, 2001a).

Relevance-based (independent) loss functions

To show that the traditional relevance based probabilistic models are special cases of risk minimization, we consider the special case where the loss function L is defined through

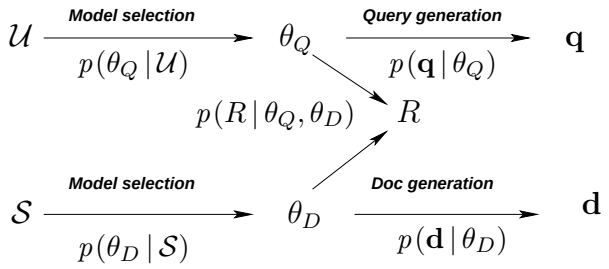

Figure 3.2: Generative model of query $\mathbf{q}$, document $\mathbf{d}$, and relevance R.

some binary relevance variable R. Specifically, we assume that for each document $\mathbf{d}_i$, there is a hidden binary relevance variable R_i that depends on θ_Q and θ_i according to $p(R_i \,|\, \theta_Q, \theta_i)$, which is interpreted as representing the true relevance status of $\mathbf{d}_i$ with respect to $\mathbf{q}$ (1 for relevant and 0 for non-relevant) (See Figure 3.2). The random variable R_i is observed when we have the user's relevance judgment on $\mathbf{d}_i$, and is unobserved otherwise. Let us assume that R_i is not observed for now. Note that because the query model θ_Q can encode detailed knowledge about the user $\mathcal{U}$, the distribution of this relevance variable can be user-specific.

Introducing the variable R into our parameter space, Equation 3.1 becomes:

$$r(\mathbf{d}|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) = \sum_{R \in \{0,1\}} \int_{\Theta} l(\mathbf{d}, R, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}})) p(R \,|\, \theta) p(\theta|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) d\theta \qquad (3.2)$$

Now let us assume that the loss function l depends on θ only through the relevance variable R. That is, let l be defined

$$l(\mathbf{d}, R, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}})) = l(R) \;\; = \;\; \begin{cases} c_0 & \text{if } R = 0 \\ c_1 & \text{if } R = 1 \end{cases}$$

where c_0 and c_1 are two cost constants, and $c_0 > c_1$ for any reasonable loss function.

From Equation 3.2, we have

$$\begin{aligned} r(\mathbf{d}|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) &= c_0 \, p(R = 0 \,|\, \mathbf{q}, \mathbf{d}) + c_1 \, p(R = 1 \,|\, \mathbf{q}, \mathbf{d}) \\ &= c_0 + (c_1 - c_0) \, p(R = 1 \,|\, \mathbf{q}, \mathbf{d}) \end{aligned}$$

This means that the risk minimization ranking criterion is now equivalent to ranking based on $p(R = 1 \,|\, \mathbf{q}, \mathbf{d})$, i.e., the probability of relevance given $\mathbf{q}$ and $\mathbf{d}$. [2] This is

[2] Since $c_0 > c_1$, a decreasing order of r is equivalent to an increasing order or $p(R = 1 \,|\, \mathbf{q}, \mathbf{d})$.

the basis of all probabilistic relevance retrieval models. Thus, we have shown that all the variants of the probabilistic relevance models reviewed in Chapter 2, Section 2.2, are all special cases of the risk minimization framework. In particular, this would cover both the classic document generation probabilistic retrieval models and the language modeling approach, which is based on query generation. See (Lafferty and Zhai, 2001a) for details of the derivation.

Distance-based (independent) loss functions

We now consider another special case, where the loss function l is proportional to a distance or similarity measure Δ between θ_Q and θ_D, i.e.,

$$l(\mathbf{d}, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}})) = c\Delta(\theta_Q, \theta_D)$$

where c is a cost constant. Intuitively, if the models θ, θ' are close/similar, then $\Delta(\theta, \theta')$ should be small.

With this loss function, from Equation 3.1, we have

$$r(\mathbf{d}|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) \propto \int_{\theta_Q} \int_{\theta_D} \Delta(\theta_Q, \theta_D)\, p(\theta_Q \mid \mathbf{q}, \mathcal{U})\, p(\theta_D \mid \mathbf{d}, \vec{\mathcal{S}}) d\theta_D\, d\theta_Q$$

This means that the risk minimization ranking criterion is now equivalent to ranking based on the expected model distance. To make this distance easier to compute, we can approximate it by its value at the posterior mode of the parameters. That is

$$\begin{aligned} r(\mathbf{d}|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) &\propto \Delta(\widehat{\theta}_Q, \widehat{\theta}_D)\, p(\theta_D = \widehat{\theta}_D \mid \mathbf{d}, \vec{\mathcal{S}})\, p(\theta_Q = \widehat{\theta}_Q \mid \mathbf{q}, \mathcal{U}) \\ &\propto \Delta(\widehat{\theta}_D, \widehat{\theta}_Q)\, p(\theta_D = \widehat{\theta}_D \mid \mathbf{d}, \vec{\mathcal{S}}) \end{aligned}$$

where

$$\begin{aligned} \widehat{\theta}_Q &= \arg\max_{\theta_Q} p(\theta_Q \mid \mathbf{q}, \mathcal{U}) \\ \widehat{\theta}_D &= \arg\max_{\theta_D} p(\theta_D \mid \mathbf{d}, \vec{\mathcal{S}}) \end{aligned}$$

Note that the factor $p(\theta_D = \widehat{\theta}_D \mid \mathbf{d}, \vec{\mathcal{S}})$ includes prior information about the document, and in general must be included when comparing the risk for different documents. This is critical when incorporating query-independent link analysis, or other extrinsic knowledge about a document. But if we further assume that $p(\theta_D = \widehat{\theta}_D \mid \mathbf{d}, \vec{\mathcal{S}})$ is the same for all $\mathbf{d}$,

so it does not affect ranking, we will have the following very general distance-based (or equivalently, similarity-based) probabilistic model:

$$r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \propto \Delta(\widehat{\theta}_D,\widehat{\theta}_Q)$$

We can view the vector space model as a special case of this general similarity model, in which $\widehat{\theta}_Q$ and $\widehat{\theta}_D$ are simply term vector parameters estimated heuristically and the distance function is the cosine or inner product measure.

3.5.2 Dependent Loss

Independent loss is not really a realistic assumption; the loss of viewing one document generally depends on the documents already viewed. For example, if the user has already seen the same document or a similar document, then the document should incur a much greater loss than if it is completely new to the user. When the independence assumption does not hold, the complexity of finding the optimal ranking makes the computation intractable. One practical solution is to use a greedy algorithm to construct a sub-optimal ranking. Specifically, we will "grow" the target ranking by choosing the document at each rank, starting from the very first rank. Suppose we already have a partially constructed ranking $\pi(1:i)$, and we are now choosing the document at rank $i+1$. Let k be a possible document index to be considered for rank $i+1$, and let $\pi(1:i,k)$ represent the ordering $(d_{\pi(1:i)^1},...,d_{\pi(1:i)^i},d_k)$. Then, the increase of risk for picking d_k at rank $i+1$ is

$$\begin{aligned}\delta(k|\pi(1:i)) &= R(\pi(1:i,k)|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) - R(\pi(1:i)|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \\ &= s_{i+1}(r(d_k|d_{\pi(1:i)^1},...,d_{\pi(1:i)^i},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) + \\ &\quad \sum_{j=1}^{i} r(d_j|d_{\pi(1:i)^1},...,d_{\pi(1:i)^{j-1}},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}))\end{aligned}$$

To extend $\pi(1:i)$, we should choose

$$\begin{aligned}k^* &= \arg\min_k \delta(k|\pi(1:i)) \\ &= \arg\min_k r(d_k|d_{\pi(1:i)^1},...,d_{\pi(1:i)^i},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}})\end{aligned}$$

Thus, at each step, we just need to evaluate

$$\delta^{'}(k|\pi(1:i)) = r(d_k|d_{\pi(1:i)^1}, ..., d_{\pi(1:i)^i}, \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) \tag{3.3}$$

and choose the k that minimizes $\delta^{'}(k|\pi(1:i))$.

This gives us a general greedy and context-dependent ranking algorithm. With this algorithm, again, we see that the "optimal" ranking does not depend on the stopping probabilities s_i. In Chapter 7, we will discuss several special cases of this algorithm, including the Maximal Marginal Relevance method (MMR) (Carbonell and Goldstein, 1998).

3.6 Discussion

3.6.1 A Decision-Theoretic View of Retrieval

Treating retrieval from a decision-theoretic view is not new. In 1970's, people were already studying how to choose and weight indexing terms from a decision-theoretic perspective (Bookstein and Swanson, 1975; Harter, 1975; Cooper and Maron, 1978). The probability ranking principle had also been justified based on optimizing the statistical decision about whether to retrieve a document (Robertson, 1977). However, the action/decision space considered in all this early work was limited to a binary decision regarding whether to retrieve a document or regarding whether to assign an index term to a document, and none of the work gave a *complete* decision-theoretic formal model for retrieval.

In the risk minimization framework, we have explicitly and formally treated the retrieval problem as a decision-making problem. The decision problem is a more general one where the action space, in principle, consists of all the possible actions that the system can take in response to a query. The scope of the decision space is a significant departure from any existing decision-theoretic treatment of retrieval. Such a general decision-theoretic view explicitly suggests that retrieval is modeled as an *interactive* process that involves cycles of a user reformulating the query and the system presenting information. We believe that this is the first time when a user variable ($\mathcal{U}$) and a document source variable ($\mathcal{S}$) have been explicitly and formally introduced in a retrieval model. Strictly speaking, $\mathcal{U}$ actually represents a user with a certain information need, thus when the same user enters another query, it should be treated as a different $\mathcal{U}$ (Robertson, 2002). However, it is not hard to imagine that we could factor out the real user and the retrieval context by using two separate

variables. Also, we have introduced a separate source variable $\mathcal{S}$ for each document, as if they are all independent. This is, again, to simplify the presentation of the framework; we can easily put constraints on these source variables, e.g., requiring all of them to be identical. The explicit introduction of $\mathcal{U}$ and $\mathcal{S}$ makes it possible for us to consider any interesting user factors and document source factors when specifying the loss function. For example, the high-precision versus high-recall preferences can be encoded by assuming a different stopping probability distribution. Interestingly, as shown formally in this chapter, when we assume an independent loss function or use a greedy algorithm to approximate the optimal ranking based on a sequentially additive loss function, the optimal solution does *not* depend on the stopping probability distribution! The redundancy among documents can be captured using a dependent loss function. Other factors such as readability of documents could also be incorporated as long as we have a model for readability.

Another major difference between the risk minimization framework and the early decision-theoretic treatment of indexing is that the early work, such as (Cooper and Maron, 1978), takes the utility in a frequency sense, i.e., the expected utility over *all possible future uses*, whereas we take a Bayesian view and consider the utility with respect to the *current* user.

The decision-theoretic view of retrieval makes it possible to model an interactive retrieval process as a sequential decision process, where the user variable $\mathcal{U}$ changes over time. Actually, if we allow the system to accept any user response, rather than just a text query, as input, then we are really going beyond retrieval toward a more general (interactive) information access system.

3.6.2 Risk Minimization and the Probability Ranking Principle

The Probability Ranking Principle (PRP) has often been taken as the foundation for probabilistic retrieval models. As stated in (Robertson, 1977), the principle is based on the following two assumptions:

> "(a) The *relevance* of a document to a request is independent of the other documents in the collection;
> (b) The *usefulness* of a relevant document to a requester may depend on the *number* of relevant documents the requester has already seen (the more he has seen, the less useful a subsequent one may be)."

Under these assumptions, the PRP provides a justification for ranking documents in descending order of probability of relevance, which can be evaluated separately for each document.

From the risk minimization framework, we have derived a general ranking formula for ranking documents based on an ascending order of the *expected risk* of a document, which can also be computed separately for each document. And we have also made two assumptions:

- Independent loss function: We assume that the loss associated with a user's viewing one document does not depend on any other documents that the user may have seen.
- Sequential browsing: We assume that, when presented with a ranked list of documents, a user would browse through the list sequentially according to the ranking.

It is interesting to note the difference and relationship between these two assumptions and the two assumptions made in (Robertson, 1977). The sequential browsing assumption is also made in (Robertson, 1977), though it is not explicitly stated (Robertson, 2002), but our independent loss assumption is stronger than the independent relevance assumption, since it is possible to define a dependent loss function based on independent relevance. Indeed, the second assumption in (Robertson, 1977) implies that the utility (or equivalently, the loss) of retrieving one document depends on the number of relevant documents that are ranked above this document, though it does not directly depend on the relevance status of any specific document. The price for this weaker assumption, however, is that the PRP is no longer guaranteed to give a ranking that is globally optimal, but only one that is optimal as a greedy algorithm. This assumption that a greedy algorithm is used to construct the optimal ranking is implicit in (Robertson, 1977), since the decision problem involves whether to retrieve one single document rather than choosing a ranking of all documents. In contrast, under our assumptions, ranking based on the expected risk can be shown to be globally optimal.

The PRP has several limitations as discussed in, e.g., (Cooper, 1994).

First, the PRP assumes that document usefulness is a binary property, but in reality it should really be a matter of degree. The independent loss ranking function that we derived does not have this limitation. Indeed, it is possible to derive the PRP in the risk minimization framework by assuming that the loss function depends only on a binary relevance variable.

Second, a ranking of documents by probability of usefulness is not always optimal. Cooper gave such an example, which essentially shows that the independent relevance assumption may not be true. Robertson discussed informally two ways to extend the PRP to address the possible dependency among documents (Robertson, 1977). Both have been captured in the risk minimization framework. The first is to go from ranking based on probability of relevance to ranking based on expected utility, which we achieve by using a loss function in the risk minimization framework. The second is essentially the greedy algorithm for ranking based on the conditional loss function. Thus, in the risk minimization framework, we provide a more formalized way to go beyond the PRP.

Indeed, as stated in (Robertson, 1977), "the estimation of probability of relevance for each document may not be the most appropriate form of prediction. The two main questions are:

- On the basis of what kinds of information can the system make the prediction?
- How should the system utilize and combine these various kinds of information?

These questions represent, indeed, the central problem of retrieval theory."

The risk minimization framework provides a formal answer to both of the questions. The information available to the system includes the user ($\mathcal{U}$), the document source ($\vec{\mathcal{S}}$), the query ($\mathbf{q}$), and the documents ($\mathcal{C}$). A "prediction" consists of selecting a subset of documents and presenting them in some way. However, one can easily imagine other possible "predictions." These factors are combined in a Bayesian decision-theoretic framework to compute an optimal prediction.

3.6.3 The Notion of Relevance

The risk minimization framework was originally motivated by the need for a general ranking function that allows us to view several different ranking criteria, including the query likelihood criterion used in the language modeling approach, within the same unified framework. As discussed in the existing literature, the retrieval problem may be decomposed into three basic components: representation of a query, representation of a document, and matching the two representations. With an emphasis on the operationality of the framework and probabilistic modeling, we make three corresponding assumptions: (1) A query can be viewed as an observation from a probabilistic query model; (2) A document can be viewed as an observation from a probabilistic document model; (3) The utility of a document with

respect to a query (i.e., the ranking criterion) is a function of the query model and document model. The flexibility in choosing different query models and document models is necessary to allow different representations of query and document. The flexibility in choosing the loss function is necessary in order to cover different notions of relevance and different ranking strategies.

As a result of these assumptions, the representation problem is essentially equivalent to that of model estimation, while the matching problem is equivalent to the estimation of the value of a utility function based on the observed query and document. In Bayesian decision theory, utility is modeled by a loss function; a loss value can be regarded as a negative utility value. Thus, we can say that the notion of relevance taken in the risk minimization framework is essentially *expected utility value*, which reflects both the user's preferences and the uncertainty of the query and document models. Such a notion of relevance is clearly more general than the traditional notion of independent topical relevance, since the utility can depend on all the factors that might affect a user's satisfaction with the system's action. For example, such factors may include a user's perception of redundancy or special characteristics of documents or the collection. This can be seen formally from the dependency of the loss function on variables such as $\mathcal{U}$, $\vec{\mathcal{S}}$, and $\mathcal{C}$.

The traditional notion of independent relevance can be obtained as a special case of this general utility notion by making an independent assumption on the loss function. Under this assumption, the optimal ranking is to rank documents based on their respective expected loss/risk. This expected risk essentially gives us an *independent* measure of relevance of a document with respect to a query. It is interesting to note that such a measure explicitly captures two different types of uncertainties. First, it is assumed that the "content" or "topic" (represented by a model) underlying a document or query is uncertain; given a document or a query, we can only estimate the model. This uncertainty reflects the system's inability to completely understand the underlying content/topic of a query or document, so it can be called "topic uncertainty." Second, even if we know the true model for the query and the document, the relevance value of the document model with respect to the query model is still uncertain and vague. This uncertainty reflects our incomplete knowledge of the user's true relevance criterion, and can be called "relevance uncertainty." The topic uncertainty is handled through computing an expectation over all possible models, while the relevance uncertainty is resolved through the specification of a concrete loss function.

As we make different approximation assumptions to simplify the computation of the risk minimization formula, we end up resolving these uncertainties in different ways. In

the general similarity-based model, for example, we resolve the topic uncertainty by picking the most likely model and rely on a similarity/distance function to measure the relevance uncertainty. The probabilistic relevance model (including the language modeling approach), however, assumes a binary relevance relationship between a query and a document, and addresses the relevance uncertainty and the topic uncertainty within one single probabilistic model. With a binary relevance relationship, a document is either relevant or non-relevant to a query, nothing in between, i.e., the different degree of relevance is not modeled; this is different from the similarity-based model.

3.6.4 Statistical Language Models for Text Retrieval

The use of language models in the risk minimization framework makes the framework quite different from other general retrieval frameworks such as the inference network; in particular, it makes the framework more *operational*. Indeed, an operational document ranking formula can always be derived by specifying three components: (1) The query model $p(\mathbf{q} \mid \theta_Q)$ and $p(\theta_Q \mid \mathcal{U})$; (2) The document model $p(\mathbf{d} \mid \theta_D)$ and $p(\theta_D \mid \mathcal{S})$; (3) The loss function. A different specification of these components leads to a different operational model.

It is thus clear that if there is any parameter involved in the retrieval formula derived from the risk minimization framework, then it would be from either the loss function or the language models for documents and queries. Parameters associated with the loss function generally represent a user's retrieval preferences, and thus should be set by the user in some meaningful way. For example, the level of redundancy tolerance could be such a parameter, and it must be set by a user, since different users may have different preferences; a high-recall preference may imply more tolerance of redundancy. On the other hand, parameters associated with the language models, in principle, can be estimated automatically. For example, in the following chapters, we will see parameters that control the smoothing of language models. Because such parameters are involved in statistical language models, it is possible to exploit statistical estimation methods to estimate the values of these parameters, thus providing a principled way for setting retrieval parameters.

Being able to estimate retrieval parameters is a major advantage of using language models for information retrieval. In Chapter 5, we will show that a new retrieval formula based on a two-stage language model smoothing method can achieve excellent retrieval performance through the completely automatic setting of retrieval/smoothing parameters.

Another advantage of using language models is that we can expect to achieve better retrieval performance through the more accurate estimation of a language model or through the use of a more reasonable language model. Thus, we will have more guidance on how to improve a retrieval model than in a traditional model. This will be demonstrated by another new retrieval model, in which feedback documents are exploited to improve the estimation of the query language model. We will show that the improved query model does indeed lead to improved retrieval performance in general.

Finally, language models are also useful for modeling the sub-topic structure of a document and the redundancy between documents. These will also be explored in the thesis as a way to achieve a non-traditional ranking of documents, e.g., to minimize redundancy or maximize sub-topic coverage.

Chapter 4

Smoothing in the Language Modeling Approach

Document language models and query language models are the building blocks of the risk minimization framework. In this chapter, we study the important problem of language model smoothing, focusing on the smoothing of document language models.

4.1 Introduction

The term *smoothing* refers to the adjustment of the maximum likelihood estimator of a language model so that it will be more accurate. At the very least, it is required to not assign a zero probability to unseen words. When estimating a language model based on a limited amount of text, such as a single document, smoothing of the maximum likelihood model is extremely important. Indeed, many language modeling techniques are centered around the issue of smoothing. In the language modeling approach to retrieval, smoothing accuracy is directly related to retrieval performance. Yet most existing research work has assumed one method or another for smoothing, and the smoothing effect tends to be mixed with that of other heuristic techniques. There has been no direct evaluation of different smoothing methods, and it is unclear how retrieval performance is affected by the choice of a smoothing method and its parameters. Thus our research questions are (1) how sensitive is retrieval performance to the smoothing of a document language model? (2) how should a smoothing method be selected, and how should its parameters be chosen?

We compare several of the most popular smoothing methods that have been developed

in speech and language processing, and study the behavior of each method.

Our study leads to several interesting and unanticipated conclusions. We find that the retrieval performance is highly sensitive to the setting of smoothing parameters. In some sense, smoothing is as important to this new family of retrieval models as term weighting is to the traditional models. Interestingly, the sensitivity pattern and query verbosity are found to be highly correlated. The performance is more sensitive to smoothing for verbose queries than for keyword queries. Verbose queries also generally require more aggressive smoothing to achieve optimal performance. This suggests that smoothing plays two different roles in the query likelihood ranking method. One role is to improve the accuracy of the estimated document language model, while the other is to accommodate the generation of common and non-informative words in a query, i.e., to explain the possible noise in a query.

The rest of the chapter is organized as follows. In Section 4.2 we discuss the language modeling approach and the connection between smoothing and some heuristics used in traditional retrieval models. In Section 4.3, we describe the three smoothing methods we evaluated. The major experiments and results are presented in Sections 4.4 through 4.7. In Section 4.8, we then present more clarification experiment results to support the dual role of smoothing and a two-stage smoothing method. Section 4.9 presents conclusions and further work.

4.2 The Language Modeling Approach

The basic idea of the language modeling approach to information retrieval can be described as follows. We assume that a query q is "generated" by a probabilistic model based on a document d. Given a query $q = q_1 q_2 ... q_n$ and a document $d = d_1 d_2 ... d_m$, we are interested in estimating the conditional probability $p(d \mid q)$, i.e., the probability that d generates the observed q. After applying the Bayes' formula and dropping a document-independent constant (since we are only interested in ranking documents), we have

$$p(d \mid q) \quad \propto \quad p(q \mid d) p(d)$$

As discussed in (Berger and Lafferty, 1999), the righthand side of the above equation has an interesting interpretation, where, $p(d)$ is our prior belief that d is relevant to any query and $p(q \mid d)$ is the query likelihood given the document, which captures how well the document "fits" the particular query q.

In the simplest case, $p(d)$ is assumed to be uniform, and so does not affect document ranking. This assumption has been taken in most existing work (Berger and Lafferty, 1999; Ponte and Croft, 1998; Ponte, 1998; Hiemstra and Kraaij, 1998; Song and Croft, 1999). In other cases, $p(d)$ can be used to capture non-textual information, e.g., the length of a document or links in a web page, as well as other format/style features of a document. In our study, we assume a uniform $p(d)$ in order to focus on the effect of smoothing. See (Miller et al., 1999) for an empirical study that exploits simple alternative priors.

With a uniform prior, the retrieval model reduces to the calculation of $p(q \mid d)$, where language modeling comes in. The language model used in most previous work is the unigram model.[1] This is the multinomial model which assigns the probability

$$p(q \mid d) \;=\; \prod_i p(q_i \mid d)$$

Clearly, the retrieval problem is now essentially reduced to a unigram language model estimation problem. In this paper we focus on unigram models only; see (Miller et al., 1999; Song and Croft, 1999) for some explorations of bigram and trigram models.

On the surface, the use of language models appears fundamentally different from vector space models with TF-IDF weighting schemes, because the unigram language model only explicitly encodes term frequency—there appears to be no use of inverse document frequency weighting in the model. However, there is an interesting connection between the language model approach and the heuristics used in the traditional models. This connection has much to do with smoothing, and an appreciation of it gives insight into the language modeling approach.

Most smoothing methods make use of two distributions, a model $p_s(w \mid \mathbf{d})$ used for "seen" words that occur in the document, and a model $p_u(w \mid \mathbf{d})$ for "unseen" words that do not. The probability of a query $\mathbf{q}$ can be written in terms of these models as follows, where $c(w; \mathbf{d})$ denotes the count of word w in $\mathbf{d}$.

$$\begin{aligned}
\log p(\mathbf{q} \mid \mathbf{d}) \;&=\; \sum_i \log p(q_i \mid \mathbf{d}) \\
&= \sum_{i:c(q_i;d)>0} \log p_s(q_i \mid \mathbf{d}) + \sum_{i:c(q_i;d)=0} \log p_u(q_i \mid \mathbf{d}) \\
&= \sum_{i:c(q_i;d)>0} \log \frac{p_s(q_i \mid \mathbf{d})}{p_u(q_i \mid \mathbf{d})} + \sum_i \log p_u(q_i \mid \mathbf{d})
\end{aligned}$$

[1]The work (Ponte and Croft, 1998) adopts something similar to, but slightly different from the standard unigram model.

The probability of an unseen word is typically taken as being proportional to the general frequency of the word, e.g., as computed using the document collection. So let us assume that $p_u(q_i \mid d) = \alpha_d \, p(q_i \mid \mathcal{C})$, where α_d is a document-dependent constant and $p(q_i \mid \mathcal{C})$ is the collection language model. Now we have

$$\log p(\mathbf{q} \mid \mathbf{d}) = \sum_{i:c(q_i;\mathbf{d})>0} \log \frac{p_s(q_i \mid \mathbf{d})}{\alpha_{\mathbf{d}} \, p(q_i \mid \mathcal{C})} + n \, \log \alpha_d + \sum_i \log p(q_i \mid \mathcal{C})$$

where n is the length of the query. Note that the last term on the righthand side is independent of the document $\mathbf{d}$, and thus can be ignored in ranking.

Now we can see that the retrieval function can actually be decomposed into two parts. The first part involves a weight for each term common to the query and document (i.e., matched terms) and the second part only involves a document-dependent constant that is related to how much probability mass will be allocated to unseen words, according to the particular smoothing method used. The weight of a matched term q_i can be identified as the logarithm of $\frac{p_s(q_i \mid \mathbf{d})}{\alpha_{\mathbf{d}} \, p(q_i \mid \mathcal{C})}$, which is directly proportional to the document term frequency, but inversely proportional to the collection frequency. The computation of this general retrieval formula can be carried out very efficiently, since it only involves a sum over the *matched* terms.

Thus, the use of $p(q_i \mid \mathcal{C})$ as a reference smoothing distribution has turned out to play a role very similar to the well-known IDF. The other component in the formula is just the product of a document-dependent constant and the query length. We can think of it as playing the role of document length normalization, which is another important technique to improve performance in traditional models. Indeed, α_d should be closely related to the document length, since one would expect that a longer document needs less smoothing and thus a smaller α_d; thus a long document incurs a greater penalty than a short one because of this term.

The connection just derived shows that the use of the collection language model as a reference model for smoothing document language models implies a retrieval formula that implements TF-IDF weighting heuristics and document length normalization. This suggests that smoothing plays a key role in the language modeling approaches to retrieval. A more restrictive derivation of the connection was given in (Hiemstra and Kraaij, 1998).

Method	$p_s(w \mid d)$	α_d	*Parameter*
Jelinek-Mercer	$(1-\lambda)\, p_{ml}(w \mid d) + \lambda\, p(w \mid \mathcal{C})$	λ	λ
Dirichlet	$\frac{c(w;d) + \mu\, p(w \mid \mathcal{C})}{\sum_w c(w;d) + \mu}$	$\frac{\mu}{\sum_w c(w;d) + \mu}$	μ
Absolute discount	$\frac{\max(c(w;d) - \delta, 0)}{\sum_w c(w;d)} + \frac{\delta \mid d \mid_u}{\mid d \mid} p(w \mid \mathcal{C})$	$\frac{\delta \mid d \mid_u}{\mid d \mid}$	δ

Table 4.1: Summary of the three primary smoothing methods compared in this paper.

4.3 Smoothing Methods

As described above, our goal is to estimate $p(w \mid \mathbf{d})$, a unigram language model based on a given document $\mathbf{d}$. The simplest method is the maximum likelihood estimate, simply given by relative counts

$$p_{ml}(w \mid \mathbf{d}) \;=\; \frac{c(w;d)}{\sum_w c(w;d)}$$

However, the maximum likelihood estimator will generally underestimate the probability of any word unseen in the document, and so the main purpose of smoothing is to assign a non-zero probability to the unseen words and improve the accuracy of word probability estimation in general.

Many smoothing methods have been proposed, mostly in the context of speech recognition tasks (Chen and Goodman, 1998). In general, all smoothing methods try to discount the probabilities of the words seen in the text, and to then assign the extra probability mass to the unseen words according to some "fallback" model. For information retrieval, it makes sense, and is very common, to exploit the collection language model as the fallback model. Following (Chen and Goodman, 1998), we assume the general form of a smoothed model to be the following:

$$p(w \mid \mathbf{d}) \;=\; \begin{cases} p_s(w \mid \mathbf{d}) & \text{if word } w \text{ is seen} \\ \alpha_d\, p(w \mid \mathcal{C}) & \text{otherwise} \end{cases}$$

where $p_s(w \mid \mathbf{d})$ is the smoothed probability of a word seen in the document, $p(w \mid \mathcal{C})$ is the collection language model, and α_d is a coefficient controlling the probability mass assigned to unseen words, so that all probabilities sum to one. In general, α_d may depend on d.

Indeed, if $p_s(w \mid \mathbf{d})$ is given, we must have

$$\alpha_d \;=\; \frac{1 - \sum_{w:c(w;d)>0} p_s(w \mid d)}{1 - \sum_{w:c(w;d)>0} p(w \mid \mathcal{C})}$$

Thus, individual smoothing methods essentially differ in their choice of $p_s(w \mid \mathbf{d})$.

A smoothing method may be as simple as adding an extra count to every word (called additive, or Laplace smoothing), or more sophisticated as in Katz smoothing, where words with different counts are treated differently. However, because a retrieval task typically requires efficient computations over a large collection of documents, our study is constrained by the efficiency of the smoothing method. We selected three representative methods that are popular and relatively efficient to implement. We excluded some well-known methods, such as Katz smoothing (Katz, 1987) and Good-Turing estimation (Good, 1953), because of the efficiency constraint. [2] Although the methods we evaluated are simple, the issues that they bring to light are relevant to more advanced methods. The three methods are described below.

The Jelinek-Mercer method. This method involves a linear interpolation of the maximum likelihood model with the collection model, using a coefficient λ to control the influence of each model.

$$p_\lambda(w \mid \mathbf{d}) \;=\; (1-\lambda)\, p_{ml}(w \mid \mathbf{d}) + \lambda\, p(w \mid \mathcal{C})$$

Thus, this is a simple mixture model (but we preserve the name of the more general Jelinek-Mercer method which involves deleted interpolation estimation of linearly interpolated n-gram models).

Bayesian smoothing using Dirichlet priors. A language model is a multinomial distribution, for which the conjugate prior for Bayesian analysis is the Dirichlet distribution with parameters

$$(\mu p(w_1 \mid \mathcal{C}),\ \mu p(w_2 \mid \mathcal{C}),\ \ldots,\ \mu p(w_n \mid \mathcal{C}))$$

Thus, the model is given by

$$p_\mu(w \mid \mathbf{d}) \;=\; \frac{c(w;d) + \mu\, p(w \mid \mathcal{C})}{\sum_w c(w;d) + \mu}$$

The Laplace method is a special case of this technique.

[2] They involve the count of words with the same frequency in a document, which is expensive to compute.

Absolute discounting. The idea of the absolute discounting method is to lower the probability of seen words by subtracting a constant from their counts (Ney et al., 1994). It is similar to the Jelinek-Mercer method, but differs in that it discounts the seen word probability by subtracting a constant instead of multiplying it by (1-λ). The model is given by

$$p_\delta(w \mid \mathbf{d}) \quad = \quad \frac{\max(c(w;d) - \delta, 0)}{\sum_w c(w;d)} + \sigma p(w \mid \mathcal{C})$$

where $\delta \in [0,1]$ is a discount constant and $\sigma = \delta \, |d|_u / |d|$, so that all probabilities sum to one. Here $|d|_u$ is the number of *unique* terms in document d, and $|d|$ is the total count of words in the document, so that $|d| = \sum_w c(w;d)$.

The three methods are summarized in Table 4.1 in terms of $p_s(w \mid \mathbf{d})$ and α_d in the general form. It is easy to see that a larger parameter value means more smoothing in all cases.

Retrieval using any of the three methods can be implemented very efficiently when the smoothing parameter is given in advance. The α's can be pre-computed for all documents at index time. The weight of a matched term w can be computed easily based on the collection language model $p(w \mid \mathcal{C})$, the query term frequency $c(w;q)$, the document term frequency $c(w;d)$, and the smoothing parameters. The scoring complexity for a query q is $O(k \ |q|)$, where $|q|$ is the query length, and k is the average number of documents in which a query term occurs. It is as efficient as scoring using a TF-IDF model.

4.4 Experimental Setup

Our goal is to study the behavior of individual smoothing methods as well as to compare different methods. As is well-known, the performance of a retrieval algorithm may vary significantly according to the testing collection used. It is generally desirable to have larger collections and more queries. We use the following five databases from TREC, including three of the largest testing collections for ad hoc retrieval, i.e., the official TREC7 ad hoc, TREC8 ad hoc, and TREC8 web track testing collections: (1) The Financial Times on disk 4, (2) FBIS on disk 5, (3) The Los Angeles Times on disk 5, (4) Disk 4 and disk 5 minus CR, used for the TREC7 and TREC8 ad hoc tasks, and (5) the TREC8 Web data. The characteristics of the databases are summarized in in Table 4.2.

The queries we use are topics 351–400 (used for the TREC7 ad hoc task), and topics 401–450 (used for the TREC8 ad hoc and web tasks). In order to study the possible in-

Collection	size	#term	#doc	avgDocLen	maxDocLen
FBIS	370MB	318,025	130,471	516	139,709
FT	446MB	259,685	209,097	394	16,021
LA	357MB	272,880	131,896	505	24,653
TREC7&8	1.36GB	747,991	527,094	474	154,322
WEB	1.26GB	1,761,265	227,724	980	332,383

Query set	min#term	avg#term	max#term
Trec7-Title	1	2.5	4
Trec7-Long	31	57.5	106
Trec8-Title	1	2.44	4
Trec8-Long	23	51.66	98

Table 4.2: Statistics of the five text collections used in our study (top), together with the queries used for testing (bottom).

teraction of smoothing and query length/type, we use two different versions of each set of queries: (1) title only, and (2) the long version (title + description + narrative). The lengths of different kinds of queries are summarized in Table 4.2. The title queries are mostly two or three key words, whereas the long queries have whole sentences and are much more verbose.

In all our experiments, the only tokenization applied is stemming with a Porter stemmer. We indexed all the words in the language, since we do not want to be biased by any artificial choice of stop words and we believe that the effects of stop word removal should be better achieved by exploiting language modeling techniques.

In Table 4.3 we give the labels used for all possible retrieval testing collections, based on the databases and queries described above.

For each smoothing method and on each testing collection, we experiment with a wide range of parameter values. In each run, the smoothing parameter is set to the same value across all queries and documents. Throughout this chapter, we follow the standard TREC evaluation procedure for ad hoc retrieval, i.e., for each topic, the performance figures are computed based on the top 1,000 documents in the retrieval results (Voorhees and Har-

Document collection	Queries			
	351-400 (Trec7)		401-450 (Trec8)	
	Title	Long	Title	Long
FBIS	fbis7T	fbis7L	fbis8T	fbis8L
FT	ft7T	ft7L	ft8T	ft8L
LA	la7T	la7L	la8T	la8L
TREC7&8	trec7T	trec7L	trec8T	trec8L
WEB	N/A		web8T	web8L

Table 4.3: Labels used for test collections.

man, 2001). To study the behavior of an individual smoothing method, we select a set of representative parameter values and examine the sensitivity of non-interpolated average precision and recall to the variation in these values. To compare smoothing methods, we first optimize the performance of each method using the non-interpolated average precision as the optimization criterion, and then compare the best runs from each method. The optimal parameter is determined by searching over the entire parameter space.[3]

4.5 Behavior of Individual Methods

In this section, we study the behavior of each smoothing method. We first derive the expected influence of the smoothing parameter on the term weighting and document length normalization implied by the corresponding retrieval function. Then we examine the sensitivity of retrieval performance by plotting the non-interpolated precision and recall at 1,000 documents against the different values of the smoothing parameter.

Jelinek-Mercer smoothing. When using the Jelinek-Mercer smoothing method with a fixed λ, we see that the parameter α_d in our ranking function (see Section 4.2) is the same for all documents, so the length normalization term is a constant. This means that the score can be interpreted as a sum of weights over each matched term. The term weight is $\log(1 + (1 - \lambda)p_{ml}(q_i|d)/(\lambda p(q_i \,|\, \mathcal{C})))$. Thus, a small λ means less smoothing and more emphasis on relative term weighting. When λ approaches zero, the weight of each term will be dominated by the term $\log(1/\lambda)$, which is term-independent, so the scoring formula will be dominated by the coordination level matching, which is simply the count of matched

[3]The search is performed in an iterative way, such that each iteration is more focused than the previous one. We stop searching when the improvement in average precision is less than 1%.

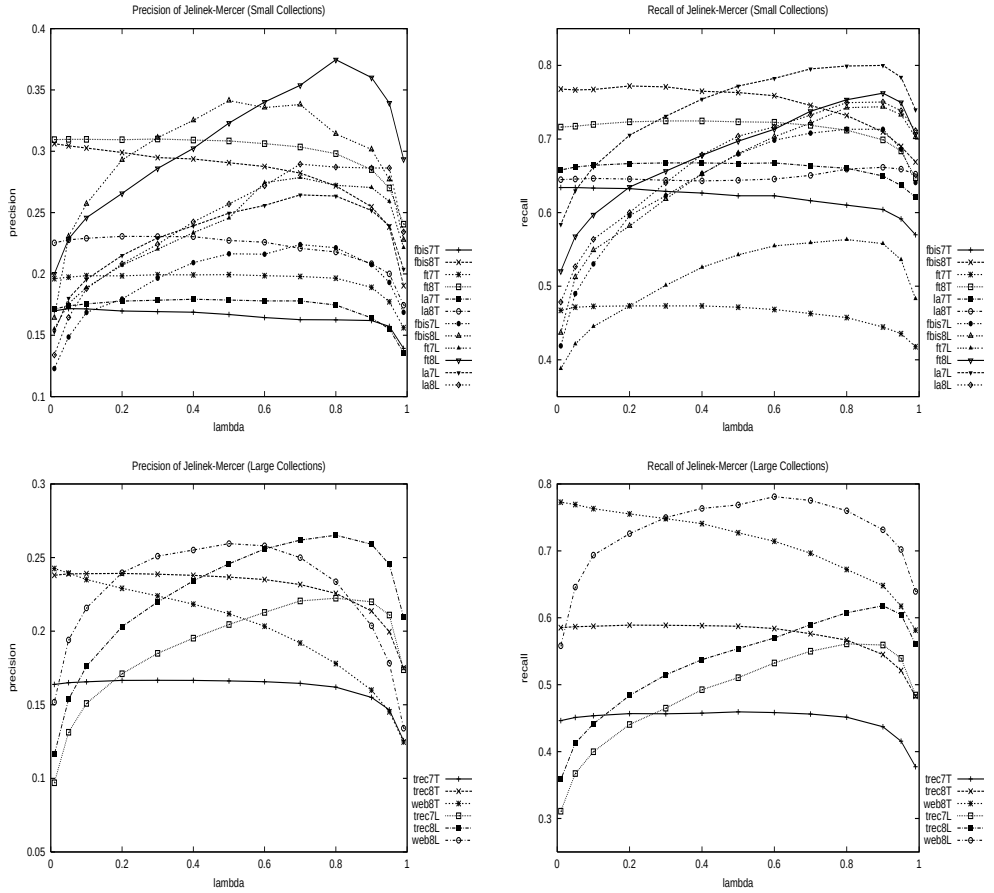

Figure 4.1: Performance of Jelinek-Mercer smoothing.

terms. This means that documents that match more query terms will all be ranked higher than those that match fewer terms, implying a *conjunctive* interpretation of the query terms. On the other hand, when λ approaches one, the weight of a term will be approximately $1 + (1 - \lambda)p_{ml}(q_i|d)/(\lambda p(q_i \,|\, \mathcal{C}))$, since $\log(1 + x) \approx x$ when x is very small. Thus, the scoring is essentially based on $\sum_i p_{ml}(q_i|d)/p(q_i \,|\, \mathcal{C})$. This means that the score will be dominated by the term with the highest weight, implying a *disjunctive* interpretation of the query terms.

The plots in Figure 4.1 show the average precision and recall for different settings of λ, for both large and small collections. It is evident that both precision and recall are much more sensitive to λ for long queries than for title queries. The web collection, however, is an exception, where performance is very sensitive to smoothing even for title queries. For title queries, the retrieval performance tends to be optimized when λ is small (around 0.1),

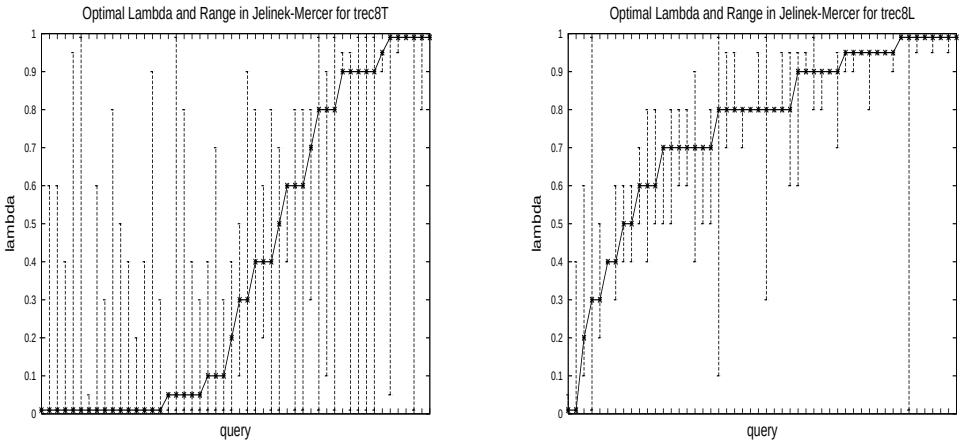

Figure 4.2: Optimal λ range for Trec8T (left) and Trec8L (right) in Jelinek-Mercer smoothing. The line shows the optimal value of λ and the bars are the optimal ranges.

whereas for long queries, the optimal point is generally higher, usually around 0.7. The difference in the optimal λ value suggests that long queries need more smoothing, and less emphasis is placed on the relative weighting of terms. The left end of the curve, where λ is very close to zero, should be close to the performance achieved by treating the query as a conjunctive Boolean query, while the right end should be close to the performance achieved by treating the query as a disjunctive query. Thus the shape of the curves in Figure 4.1 suggests that it is appropriate to interpret a title query as a conjunctive Boolean query, while a long query is more close to a disjunctive query, which makes sense intuitively.

The performance sensitivity can be seen more clearly from the per-topic plot of the optimal range of λ shown in Figure 4.2. The optimal range is defined as the maximum range of λ values that deviate from the optimal average precision by no more than 0.01.

Dirichlet priors. When using the Dirichlet prior for smoothing, we see that the α_d in the retrieval formula is document-dependent. It is smaller for long documents, so it can be interpreted as a length normalization component that penalizes long documents. The weight for a matched term is now $\log(1 + c(q_i; d)/(\mu p(q_i \mid \mathcal{C})))$. Note that in the Jelinek-Mercer method, the term weight has a document length normalization implicit in $p_s(q_i \mid d)$, but here the term weight is affected by only the raw counts of a term, not the length of the document. After rewriting the weight as $\log(1 + |d|p_{ml}(q_i \mid d)/(\mu p(q_i \mid \mathcal{C})))$ we see that $|d|/\mu$ is playing the same role as $(1 - \lambda)/\lambda$, but it differs in that it is document-dependent. The relative weighting of terms is emphasized when we use a smaller μ. Just as in Jelinek-Mercer, it can also be shown that when μ approaches zero, the scoring formula is dominated

by the count of matched terms. As μ gets very large, however, it is more complicated than Jelinek-Mercer due to the length normalization term, and it is not obvious what terms would dominate the scoring formula.

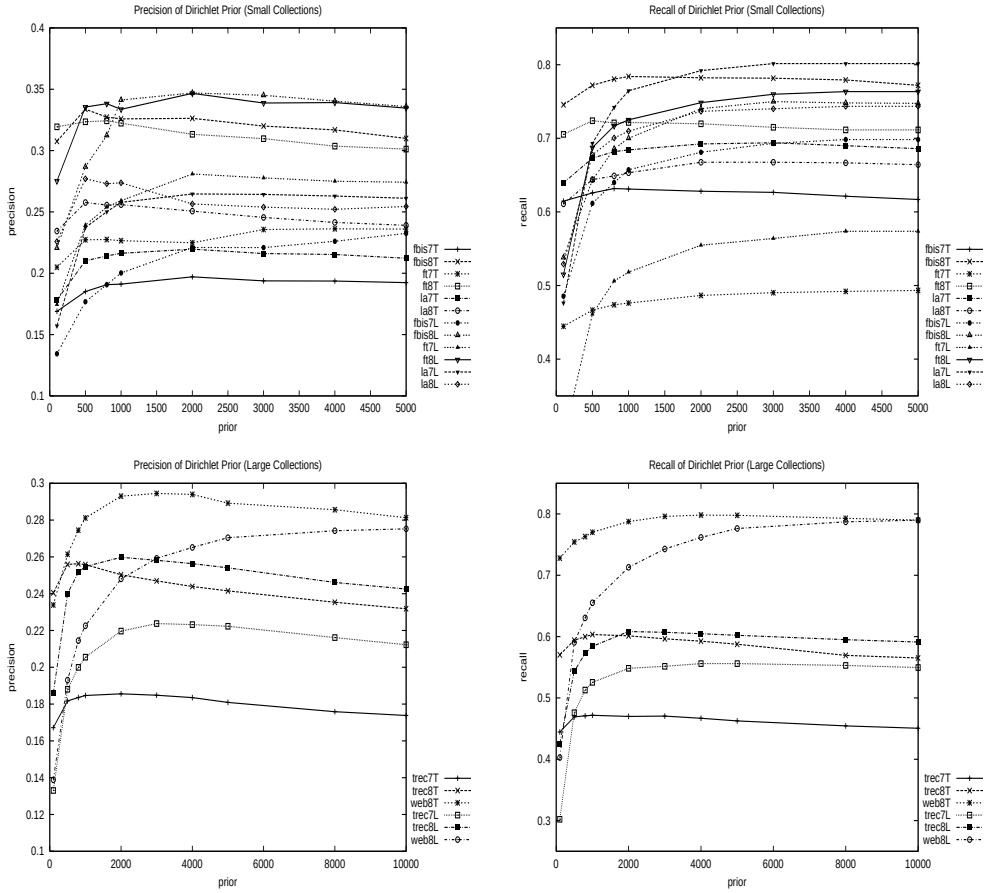

Figure 4.3: Performance of Dirichlet smoothing.

The plots in Figure 4.3 show the average precision and recall for different settings of the prior sample size μ. It is again clear that both precision and recall are much more sensitive to μ for long queries than for title queries, especially when μ is small. However, the *optimal* value of μ does not seem to be much different for title queries and long queries. It still tends to be slightly larger for long queries, but the difference is not as large as in Jelinek-Mercer. The optimal prior μ seems to vary from collection to collection, though in most cases, it is around 2,000. The tail of the curves is generally flat.

Absolute discounting. The term weighting behavior of the absolute discounting method is a little more complicated. Obviously, here α_d is also document sensitive. It is larger

for a document with a flatter distribution of words, i.e., when the count of unique terms is relatively large. Thus, it penalizes documents with a word distribution highly concentrated on a small number of words. The weight of a matched term is $\log(1 + (c(q_i; d) - \delta)/(\delta|d|_u p(q_i \,|\, \mathcal{C})))$. The influence of δ on relative term weighting depends on $|d|_u$ and $p(\cdot \,|\, \mathcal{C})$, in the following way. If $|d|_u \, p(w \,|\, \mathcal{C}) > 1$, a larger δ will make term weights flatter, but otherwise, it will actually make the term weight more skewed according to the count of the term in the document. Thus, a larger δ will amplify the weight difference for rare words, but flatten the difference for common words, where the "rarity" threshold is $p(w \,|\, \mathcal{C}) < 1/|d|_u$.

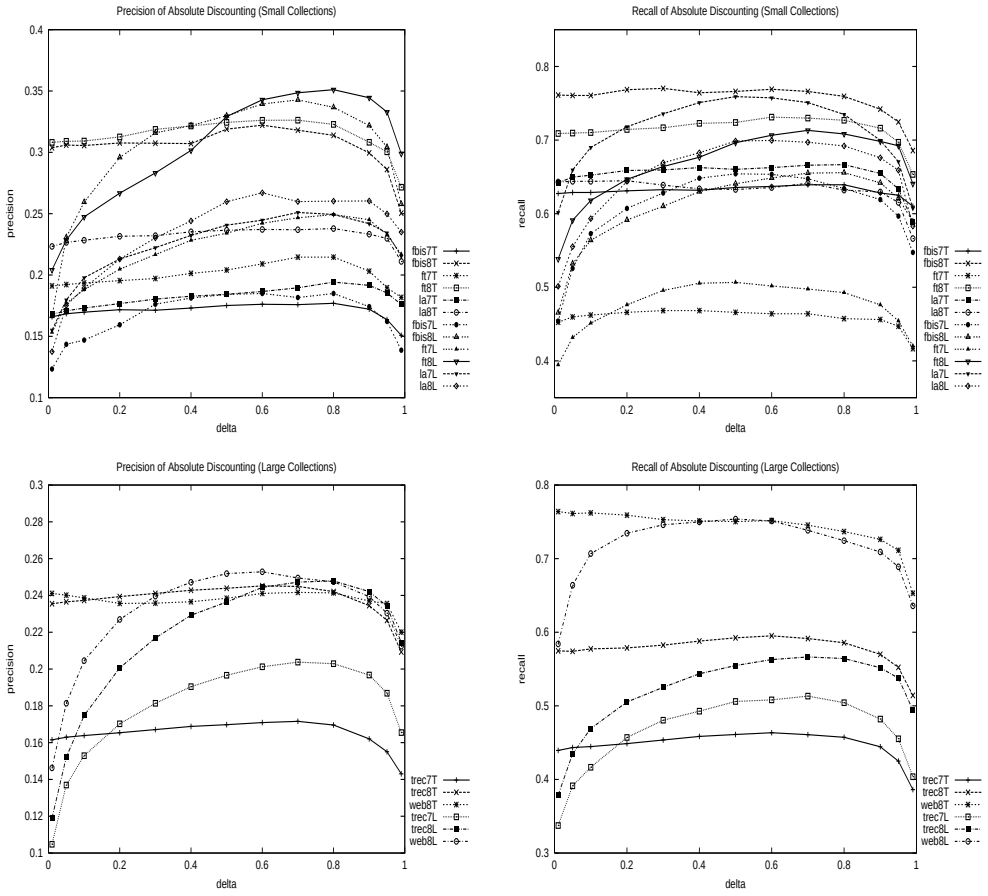

Figure 4.4: Performance of absolute discounting.

The plots in Figure 4.4 show the average precision and recall for different settings of the discount constant δ. Once again it is clear that both precision and recall are much more sensitive to δ for long queries than for title queries. Similar to Bayesian smoothing, but different from Jelinek-Mercer smoothing, the optimal value of δ does not seem to be much

different for title queries and long queries. Indeed, the optimal value of δ tends to be around 0.7. This is true not only for both title queries and long queries, but also across all testing collections.

The behavior of each smoothing method indicates that, in general, the performance of long verbose queries is much more sensitive to the choice of smoothing parameters than that of concise title queries. Inadequate smoothing hurts the performance more severely in the case of long and verbose queries. This suggests that smoothing plays a more important role for long verbose queries than for concise title queries. One interesting observation is that the web collection behaves quite differently than other databases for Jelinek-Mercer and Dirichlet smoothing, but not for absolute discounting. In particular, the title queries performed much better than the long queries on the web collection when using Dirichlet prior. Further analysis and evaluation are needed to understand this observation.

4.6 Interpolation vs. Backoff

The three methods that we have described and tested so far belong to the category of interpolation-based methods, in which we discount the counts of the seen words and the extra counts are *shared* by both the seen words and unseen words. One problem with this approach is that a high count word may actually end up with more than its actual count in the document, if the fallback model gives the word a very high probability. An alternative smoothing strategy is "backoff." Here the main idea is to trust the maximum likelihood estimate for high count words, and to discount and redistribute mass only for the less common terms. As a result, it differs from the interpolation strategy in that the extra counts are primarily used for unseen words. The Katz smoothing method is a well-known backoff method (Katz, 1987). The backoff strategy is very popular in speech recognition tasks.

Following (Chen and Goodman, 1998), we implemented a backoff version of all three interpolation-based methods, which is derived as follows. Recall that in all three methods, $p_s(w)$ is written as the sum of two parts: (1) a discounted maximum likelihood estimate, which we denote by $p_{dml}(w)$; and (2) a collection language model term, i.e., $\alpha_d\, p(w \mid \mathcal{C})$. If we use only the first term for $p_s(w)$ and renormalize the probabilities, we will have a smoothing method that follows the backoff strategy. It is not hard to show that if an interpolation-based smoothing method is characterized by $p_s(w) = p_{dml}(w) + \alpha_d\, p(w \mid \mathcal{C})$ and $p_u(w) = \alpha_d\, p(w \mid \mathcal{C})$, then the backoff version is given by $p_s^{'}(w) = p_{dml}(w)$ and $p_u^{'}(w) = \frac{\alpha_d\, p(w \mid \mathcal{C})}{1 - \sum_{i:c(w_i;d)>0} p(w_i \mid \mathcal{C})}$. The form of the ranking formula and the smoothing parame-

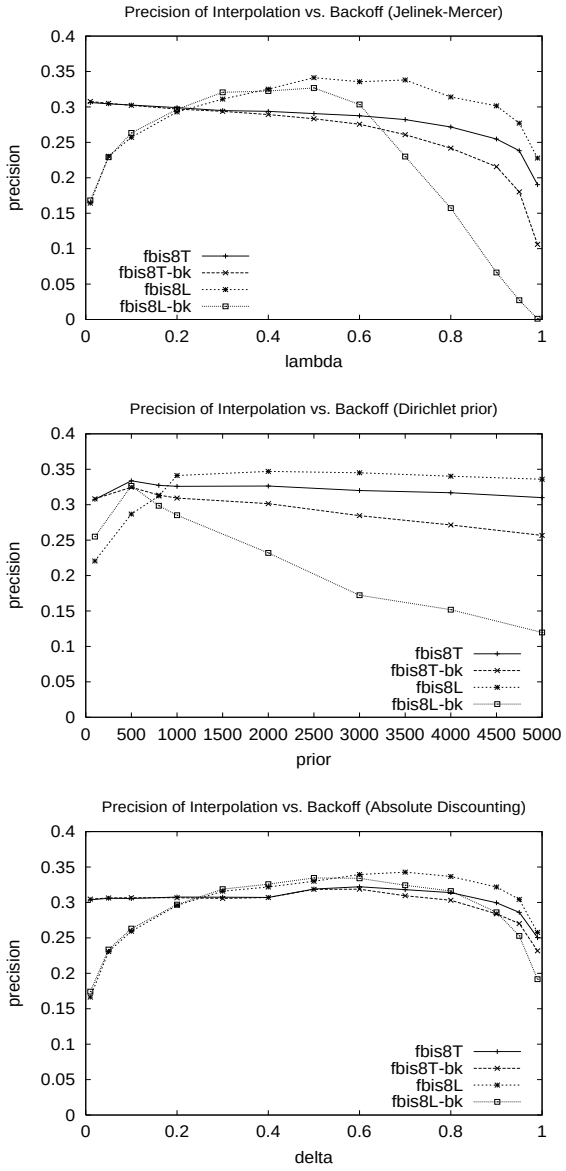

Figure 4.5: Interpolation versus backoff for Jelinek-Mercer (top), Dirichlet smoothing (middle), and absolute discounting (bottom).

ters remain the same. It is easy to see that the α_d in the backoff version differs from that in the interpolation version by a document-dependent term which further penalizes long documents. The weight of a matched term due to backoff smoothing has a much wider range of values ($(-\infty, +\infty)$) than that for interpolation ($(0, +\infty)$). Thus, analytically, the backoff version tends to do term weighting and document length normalization more aggressively than the corresponding interpolated version.

The backoff strategy and the interpolation strategy are compared for all three methods using the FBIS database and topics 401-450 (i.e., fbis8T and fbis8L). The results are shown in Figure 4.5. We find that compared with interpolation, the backoff performance is more sensitive to the smoothing parameter, especially for Jelinek-Mercer and the Dirichlet prior. The difference is clearly less significant in the absolute discounting method, and this may be due to its lower upper bound ($\frac{|d|_u}{|d|}$) for the original α_d, which restricts the aggressiveness in penalizing long documents. In general, the backoff strategy yields worse performance than the interpolation strategy, and only comes close to it when α_d approaches zero, which is expected, since analytically, we know that when α_d approaches zero, the difference between the two strategies will diminish.

4.7 Comparison of Methods

To compare the three smoothing methods, we select a best run (in terms of non-interpolated average precision) for each interpolation-based method on each testing collection and compare the non-interpolated average precision, precision at 10 documents, and precision at 20 documents of the selected runs. The results are shown in Table 4.4 and Table 4.5 for titles and long queries respectively.

For title queries, there seems to be a clear order among the three methods in terms of all three precision measures: the Dirichlet prior is better than absolute discounting, which is better than Jelinek-Mercer. Indeed, the Dirichlet prior has the best average precision in all cases but one, and its overall average performance is significantly better than that of the other two methods according to the Wilcoxin test. It performed extremely well on the Web collection, significantly better than the other two. This good performance is relatively insensitive to the choice of μ. Indeed, many non-optimal Dirichlet runs are also significantly better than the optimal runs for Jelinek-Mercer and absolute discounting.

For long queries, there is also a partial order. On average, Jelinek-Mercer is better than the Dirichlet prior and absolute discounting by all three precision measures, but its average precision is almost identical to that of Dirichlet and only the precision at 20 documents is significantly better than the other two methods according to the Wilcoxin test. Both Jelinek-Mercer and the Dirichlet prior clearly have a better average precision than absolute discounting.

When comparing each method's performance on different types of queries in Table 4.6, we see that the three methods all perform better on long queries than on title queries (except

Collection	Method	Parameter	Avg. Prec.	Prec@10	Prec@20
fbis7T	JM	$\lambda = 0.05$	0.172	**0.284**	0.220
	Dir	$\mu = 2,000$	**0.197**	0.282	**0.238**
	Dis	$\delta = 0.8$	0.177	**0.284**	0.233
ft7T	JM	$\lambda = 0.5$	0.199	0.263	0.195
	Dir	$\mu = 4,000$	**0.236**	**0.283**	**0.213**
	Dis	$\delta = 0.8$	0.215	0.271	0.196
la7T	JM	$\lambda = 0.4$	0.179	0.238	0.205
	Dir	$\mu = 2,000$	**0.220***	**0.294***	**0.233**
	Dis	$\delta = 0.8$	0.194	0.268	0.216
fbis8T	JM	$\lambda = 0.01$	0.306	0.344	0.282
	Dir	$\mu = 500$	**0.334**	**0.367**	**0.292**
	Dis	$\delta = 0.5$	0.319	0.363	0.288
ft8T	JM	$\lambda = 0.3$	0.310	0.359	0.283
	Dir	$\mu = 800$	0.324	**0.367**	**0.297**
	Dis	$\delta = 0.7$	**0.326**	**0.367**	0.296
la8T	JM	$\lambda = 0.2$	0.231	0.264	0.211
	Dir	$\mu = 500$	**0.258**	0.271	0.216
	Dis	$\delta = 0.8$	0.238	**0.282**	**0.224**
trec7T	JM	$\lambda = 0.3$	0.167	0.366	0.315
	Dir	$\mu = 2,000$	**0.186***	**0.412**	**0.342**
	Dis	$\delta = 0.7$	0.172	0.382	0.333
trec8T	JM	$\lambda = 0.2$	0.239	0.438	0.378
	Dir	$\mu = 800$	**0.256**	0.448	0.398
	Dis	$\delta = 0.6$	0.245	**0.466**	**0.406**
web8T	JM	$\lambda = 0.01$	0.243	0.348	0.293
	Dir	$\mu = 3,000$	**0.294***	**0.448***	**0.374***
	Dis	$\delta = 0.7$	0.242	0.370	0.323
Average	JM	—	0.227	0.323	0.265
	Dir	—	**0.256***	**0.352***	**0.289***
	Dis	—	0.236	0.339	0.279

Table 4.4: Comparison of smoothing methods on title queries. The best performance is shown in bold. The cases where the best performance is significantly better than the next best according to the Wilcoxin signed rank test at the level of 0.05 are marked with a star (*). JM denotes Jelinek-Mercer, Dir denotes the Dirichlet prior, and Dis denotes smoothing using absolute discounting.

that Dirichlet prior performs worse on long queries than title queries on the web collection). The improvement in average precision is statistically significant for all three methods according to the Wilcoxin test. But the performance increase is most significant for Jelinek-Mercer. Indeed, Jelinek-Mercer is the worst for title queries, but the best for long queries. It appears that Jelinek-Mercer is much more effective when queries are long and more verbose.

Since the Trec7&8 database differs from the combined set of FT, FBIS, LA by only the Federal Register database, we can also compare the performance of a method on the

Collection	Method	Parameter	Avg. Prec.	Prec@10	Prec@20
fbis7L	JM	$\lambda = 0.7$	0.224	**0.339**	**0.279**
	Dir	$\mu = 5,000$	**0.232**	0.313	0.249
	Dis	$\delta = 0.6$	0.185	0.321	0.259
ft7L	JM	$\lambda = 0.7$	0.279	**0.331**	0.244
	Dir	$\mu = 2,000$	**0.281**	0.329	**0.248**
	Dis	$\delta = 0.8$	0.249	0.317	0.236
la7L	JM	$\lambda = 0.7$	0.264	0.350	**0.286**
	Dir	$\mu = 2,000$	**0.265**	**0.354**	0.285
	Dis	$\delta = 0.7$	0.251	0.340	0.279
fbis8L	JM	$\lambda = 0.5$	0.341	0.349	0.283
	Dir	$\mu = 2,000$	**0.347**	0.349	**0.290**
	Dis	$\delta = 0.7$	0.343	**0.356**	0.274
ft8L	JM	$\lambda = 0.8$	**0.375**	**0.427**	**0.320**
	Dir	$\mu = 2,000$	0.347	0.380	0.297
	Dis	$\delta = 0.8$	0.351	0.398	0.309
la8L	JM	$\lambda = 0.7$	**0.290***	**0.296**	**0.238**
	Dir	$\mu = 500$	0.277	0.282	0.231
	Dis	$\delta = 0.6$	0.267	0.287	0.222
trec7L	JM	$\lambda = 0.8$	0.222	**0.476**	**0.401**
	Dir	$\mu = 3,000$	**0.224**	0.456	0.383
	Dis	$\delta = 0.7$	0.204	0.460	0.396
trec8L	JM	$\lambda = 0.8$	**0.265**	0.504	**0.434**
	Dir	$\mu = 2,000$	0.260	0.484	0.4
	Dis	$\delta = 0.8$	0.248	**0.518**	0.428
web8L	JM	$\lambda = 0.5$	0.259	**0.422**	**0.348**
	Dir	$\mu = 10,000$	**0.275**	0.410	0.343
	Dis	$\delta = 0.6$	0.253	0.414	0.333
Average	JM	—	**0.280**	**0.388**	**0.315***
	Dir	—	0.279	0.373	0.303
	Dis	—	0.261	0.379	0.304

Table 4.5: Comparison of smoothing methods on long queries. The best performance is shown in bold. The cases where the best performance is significantly better than the next best according to the Wilcoxin signed rank test at the level of 0.05 are marked with a star (*). JM denotes Jelinek-Mercer, Dir denotes the Dirichlet prior, and Dis denotes smoothing using absolute discounting.

three smaller databases with that on the large one. We find that the non-interpolated average precision on the large database is generally much worse than that on the smaller ones, and is often similar to the worst one among all three small databases. However, the precision at 10 (or 20) documents on large collections is significantly better than that on small collections. For both title queries and long queries, the *relative* performance of each method tends to remain the same when we merge the databases. Interestingly, the optimal setting for the smoothing parameters seems to stay within a similar range when databases are merged.

The strong correlation between the effect of smoothing and the type of queries is some-

Smoothing	Query	avg. pr	pr@10doc	pr@20doc
Jelinek-Mercer	Title	0.227	0.323	0.265
	Long	0.280	0.388	0.315
	improve	***23.3%**	***20.1%**	***18.9%**
Dirichlet	Title	0.256	0.352	0.289
	Long	0.279	0.373	0.303
	improve	***9.0%**	**6.0%**	**4.8%**
Absolute Disc.	Title	0.236	0.339	0.279
	Long	0.261	0.379	0.304
	Improve	***10.6%**	**11.8%**	**9.0%**

Table 4.6: Comparing long queries with short queries. Star (*) indicates the improvement is statistically significant according to the Wilcoxin signed rank test at the level of 0.05.

how unexpected. If the purpose of smoothing is only to improve the accuracy in estimating a unigram language model based on a document, then the effect of smoothing should be more affected by the characteristics of documents and the collection, and should be relatively insensitive to the type of queries. But the results above suggest that this is not the case. Indeed, the effect of smoothing clearly interacts with some of the query factors. To understand whether it is the length or the verbosity of the long queries that is responsible for such interactions, we design experiments to further examine these two query factors (i.e., the length and verbosity). The details are reported in the next section.

4.8 The Dual Role of Smoothing

In this section, we report the results of further experiments with four different types of queries – short keyword, long keyword, short verbose, and long verbose. We compare how they each behave with respect to smoothing to clarify which query factor has caused the strong interaction with smoothing. We will show that the high sensitivity of retrieval performance to smoothing is caused by the presence of common words in the query, independent of the query length. This suggests that smoothing plays two different roles – to make the estimated document language model more accurate and to "explain" the non-informative words in the query. Accordingly, we propose a two-stage smoothing strategy that decouples these two roles of smoothing, and thus facilitates the setting of smoothing parameters.

The four types of queries used in our experiments are generated from TREC topics 1-150. These 150 topics are special because they all have a "concept" field, which contains a list of keywords related to the topic. These keywords serve well as the "long keyword" version of our queries. Figure 4.6 shows an example of such a topic (topic 52).

```
Title:  South African Sanctions
Description: Document discusses sanctions against  South Africa.
Narrative:
   A relevant document will discuss any aspect of South African sanctions,
such as: sanctions declared/proposed by a country against the South
African government in response to its apartheid policy, or in response to
pressure by an individual, organization or another country; international
sanctions against Pretoria imposed by the United Nations; the effects of
sanctions against S. Africa; opposition to sanctions; or, compliance with
sanctions by a company.  The document will identify the sanctions instituted
or being considered, e.g., corporate disinvestment, trade ban, academic
boycott, arms embargo.

Concepts:

1. sanctions, international sanctions, economic sanctions
2. corporate exodus, corporate disinvestment, stock divestiture, ban on
   new investment, trade ban, import ban on South African diamonds,
   U.N. arms embargo, curtailment of defense contracts, cutoff of
   nonmilitary goods, academic boycott, reduction of cultural ties
3. apartheid, white domination, racism
4. antiapartheid, black majority rule
5. Pretoria
```

Figure 4.6: Example topic, number 52. The keywords are used as the "long keyword" version of our queries.

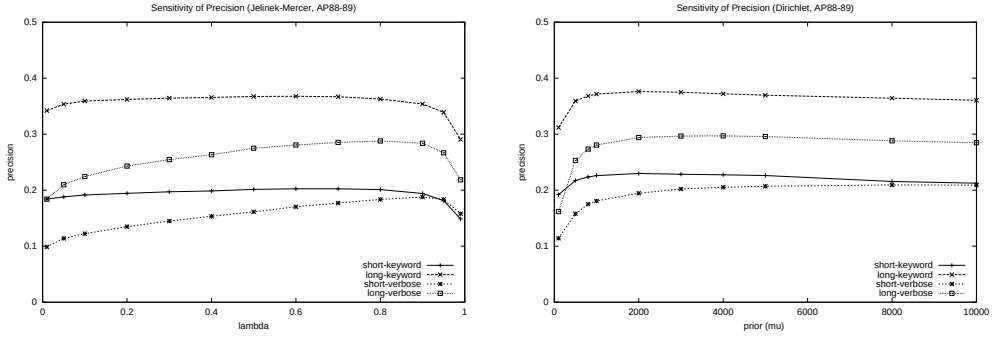

Figure 4.7: Sensitivity of average precision for Jelinek-Mercer smoothing (left) and Dirichlet prior smoothing (right) on AP88-89.

We used all of the 150 topics and generated the four versions of queries in the following way:

1. short keyword: Using only the title of the topic description (usually a noun phrase)[4]

[4]Occasionally, a few function words were manually excluded, in order to make the queries purely keyword-based.

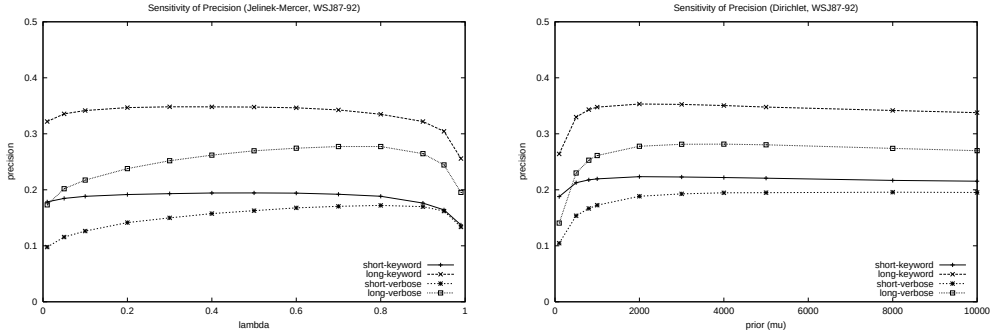

Figure 4.8: Sensitivity of average precision for Jelinek-Mercer smoothing (left) and Dirichlet prior smoothing (right) on WSJ87-92.

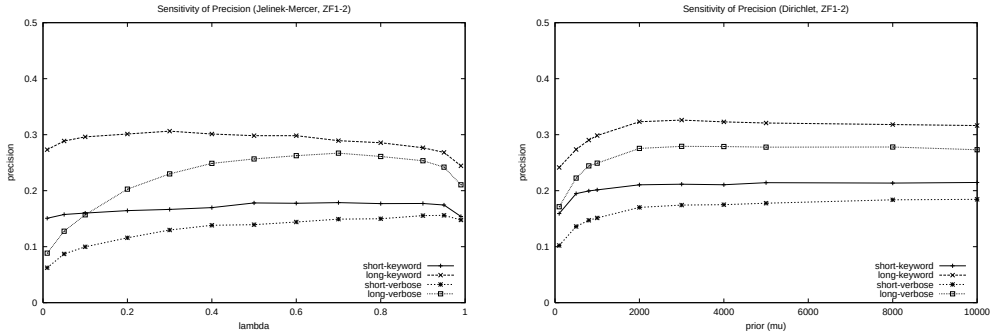

Figure 4.9: Sensitivity of average precision for Jelinek-Mercer smoothing (left) and Dirichlet prior smoothing (right) on ZF1-2.

2. short verbose: Using only the description field (usually one sentence).

3. long keyword: Using the concept field (about 28 keywords on average).

4. long verbose: Using the title, description and the narrative field (more than 50 words on average).

Thus, the verbose queries consist of one or more natural language sentences, whereas the keyword queries consist of one of more content-carrying noun phrases.

The relevance judgments available for these 150 topics are mostly on the documents in TREC disk 1 and disk 2. In order to observe any possible difference in smoothing caused by the types of documents, we partition the documents in disks 1 and 2 and use the three largest subsets of documents, accounting for a majority of the relevant documents for our queries. The three databases are AP88-89, WSJ87-92, and ZIFF1-2, each about 400MB–

500MB in size. The queries without relevance judgments for a particular database were ignored for all of the experiments on that database. Four queries do not have judgments on AP88-89, and 49 queries do not have judgments on ZIFF1-2. Preprocessing of the documents is minimized; only a Porter stemmer is used, and no stop words are removed. Combining the four types of queries with the three databases gives us a total of 12 different testing collections.

To better understand the interaction between different query factors and smoothing, we studied the sensitivity of retrieval performance to smoothing on each of the four different types of queries. For both Jelinek-Mercer and Dirichlet smoothing, on each of our 12 testing collections we vary the value of the smoothing parameter and record the retrieval performance at each parameter value. The results are plotted in Figures 4.7, 4.8, and 4.9. In each case, we plotted how the average precision varies according to different values of the smoothing parameter.

From these figures, we easily see that the two types of keyword queries behave similarly, as do the two types of verbose queries. The retrieval performance is generally much less sensitive to smoothing in the case of the keyword queries than for the verbose queries, whether long or short. Therefore, the sensitivity is much more correlated with the verbosity of the query than with the length of the query. Indeed, the short verbose queries are clearly more sensitive than the long keyword queries for Jelinek-Mercer smoothing. In general, insufficient smoothing is more harmful for verbose queries than for keyword queries. This suggests that smoothing is responsible for "explaining" the common words in a query.

We also see a consistent order of performance among the four types of queries. As expected, long keyword queries are the best and short verbose queries are the worst. Long verbose queries are worse than long keyword queries, but better than short keyword queries, which are better than the short verbose queries. This appears to suggest that queries with only (presumably good) keywords tend to perform better than more verbose queries. Also, longer queries are generally better than short queries.

The interaction patterns between query verbosity and smoothing that we observed in all these experiments suggest that smoothing actually plays two different roles in the query likelihood retrieval method. One role is to improve the accuracy of the estimated documents language model, and can be referred to as the *estimation* role. The other is to "explain" the common and non-informative words in a query, and can be referred to as the role of *query modeling*. Indeed, this second role is explicitly implemented with a two-state HMM in (Miller et al., 1999). This role is also well-supported by the connection of smoothing and

IDF weighting derived in Section 4.2. Intuitively, more smoothing would decrease the "discrimination power" of common words in the query, because all documents will rely more on the collection language model to generate the common words. The need for query modeling may also explain why the backoff smoothing methods have not worked as well as the corresponding interpolation-based methods – it is because they do not allow for query modeling.

For any particular query, the observed effect of smoothing is likely a combination of both roles of smoothing. However, for a concise keyword query, the effect can be expected to be much more dominated by the estimation role, since such a query has few or no non-informative common words. On the other hand, for a verbose query, the role of query modeling will have more influence, for such a query generally has a high ratio of non-informative common words. Thus, the results reported in Section 4.7 show that the Dirichlet prior method performs the best on concise title queries, suggesting that it is good for the estimation role, and that Jelinek-Mercer performs the worst for title queries, but the best for long verbose queries, suggesting that Jelinek-Mercer is good for the role of query modeling. Intuitively this also makes sense, as the Dirichlet prior adapts to the length of documents naturally, which is desirable for the estimation role, while in Jelinek-Mercer, we set a fixed smoothing parameter across all documents, which is necessary for query modeling.

These observations on the dual role of smoothing empirically suggest a two-stage smoothing method, which decouples the two different roles. In the first stage, a document language model is smoothed using the Dirichlet prior (to implement the estimation role). In the second stage, it is further smoothed using Jelinek-Mercer (to implement the query modeling role). Since Jelinek-Mercer smoothing can be regarded as a simple mixture model, the second stage is essentially to introduce a generative mixture model for queries, which involves mixing the document language model with a "query background" language model that can "generate" the common words in a query. In Chapter 5, we formally derive this two-stage smoothing method from the risk minimization framework, and study it in more detail.

4.9 Conclusions and Further Work

We have studied the problem of language model smoothing in the context of the query likelihood retrieval method. By rewriting the query likelihood retrieval formula using a smoothed document language model, we derived a general retrieval formula where the

smoothing of the document language model can be interpreted in terms of several heuristics used in traditional models, including TF-IDF weighting and document length normalization. We then examined three popular interpolation-based smoothing methods (the Jelinek-Mercer method, Dirichlet priors, and absolute discounting), as well as their backoff versions, and evaluated them using several large and small TREC retrieval testing collections. We found that the retrieval performance was generally sensitive to the smoothing parameters, suggesting that an understanding and appropriate setting of smoothing parameters is very important in the language modeling approach.

While our results are not completely conclusive as to which smoothing method is the best, we have made several interesting observations that help us understand each of the methods better. On all the test collections, the Jelinek-Mercer method generally performs well, but tends to perform much better for long queries than for title queries. The optimal value of λ has a strong correlation with the query type. For concise title queries, the optimal value is generally very small (around 0.1), while for long verbose queries, the optimal value is much larger (around 0.7). The Dirichlet prior method generally performs well, but tends to perform much better for concise title queries than for long verbose queries. The optimal value of μ appears to have a wide range (500-10000) and usually is around 2,000. A large value is "safer," especially for long verbose queries. The absolute discounting method performs well on concise title queries, but not very well on long verbose queries. Interestingly, there is little variation in the optimal value for δ (generally around 0.7 in all cases).

While used successfully in speech recognition, the backoff strategy did not work well for retrieval in our evaluation. All interpolated versions perform significantly better than their backoff versions.

A very interesting observation is that the effect of smoothing is strongly correlated with the type of queries. The performance is generally more sensitive to smoothing for verbose queries than for keyword queries. Verbose queries also generally require more aggressive smoothing to achieve optimal performance. This suggests that smoothing plays two different roles in the query likelihood retrieval method. One role is to improve the accuracy of the estimated document language model (estimation role), while the other is to accommodate generation of non-informative common words in the query (query modeling role). The results further suggest that the Dirichlet prior may be good for the estimation role, while Jelinek-Mercer may be good for the query modeling role. This then motivates us to propose a two-stage smoothing strategy that combines the Dirichlet prior with Jelinek-

Mercer and explicitly decouples the two roles of smoothing (the Dirichlet prior for the estimation role and Jelinek-Mercer for the second). This two-stage smoothing method is further studied in more detail in the next chapter.

There are several interesting future research directions. First, we could evaluate other more sophisticated smoothing algorithms, such as Good-Turing smoothing (Good, 1953), Katz smoothing (Katz, 1987), Kneser-Ney smoothing (Kneser and Ney, 1995)). Given the success of the Dirichlet prior smoothing method, it would also be interesting to explore smoothing with a hierarchical Bayesian model, e.g., as explored in (MacKay and Peto, 1995). Second, it is very important to study how to set the smoothing parameters automatically. In all the reported experiments, we exhaustively search the space of the parameter values in order to study sensitivity. In practice, this would be impossible, and we need to have guidance on how to set them. In general, it would be very interesting to explore the possibility of training/estimating these parameters using the query, the collection, and the past relevance judgments that are available to us. In the next chapter, we present method for estimating parameters in a two-stage smoothing method, which is one step in this important direction.

Chapter 5

Two-stage Language Models

The query likelihood retrieval method discussed in the previous chapter is based on the original language modeling approach proposed in (Ponte and Croft, 1998). It involves a two-step scoring procedure. First, estimate a document language model for each document, and, second, compute the query likelihood using the estimated document language model directly. In this chapter, we derive a family of two-stage language models using the risk minimization framework. These models generalize this two-step procedure by introducing a query generative model (Zhai and Lafferty, 2002). As a result, in the second step, instead of using the estimated document model directly, we use the query generative model, which is based on the estimated document model, to compute the query likelihood.

From the viewpoint of smoothing, we can regard such a two-stage language modeling approach as involving a two-stage smoothing of the original document model. The first-stage smoothing happens when we estimate the document language model, and the second stage is implemented through the query generative model. The two-stage smoothing method suggested in the previous chapter can be easily obtained as a special case of the two-stage language models where we use a Bayesian approach to estimate the document language model and a mixture model for query generation.

An important advantage of the two-stage language models is that they explicitly capture the different influences of the query and document collection on smoothing. It is known that the optimal setting of retrieval parameters generally depends on both the document collection and the query; decoupling the influence of the query from that of documents makes it easier to estimate smoothing parameters independently according to different documents and different queries. We present methods for estimating the two parameters involved in the

two-stage smoothing method automatically. Extensive evaluation shows that the two-stage smoothing method, with automatically estimated parameter settings, can achieve excellent retrieval performance that is either very close to, or better than, the ideal retrieval performance using single-stage smoothing methods.

5.1 Introduction

It is well known that the optimal settings of retrieval parameters generally depend on both the document collection and the query. For example, specialized term weighting for short queries was studied in (Kwok and Chan, 98). Salton and Buckley studied many different term weighting methods used in the vector space retrieval model; their recommended methods strongly depend on the type of the query and the characteristics of the document collection (Salton and Buckley, 1988). It has been a great challenge to find the optimal settings of retrieval parameters automatically and adaptively accordingly to the characteristics of the collection and queries, and empirical parameter tuning seems to be inevitable in order to achieve good retrieval performance. This is evident in the large number of parameter tuning experiments reported in virtually every paper published in the TREC proceedings (Voorhees and Harman, 2001).

The need for empirical parameter tuning is due in part from the fact that, in traditional retrieval models, such as the vector space model (Salton et al., 1975a) and the BM25 retrieval model (Robertson et al., 1995), the retrieval parameters have almost always been introduced heuristically. The lack of a direct modeling of queries and documents makes it hard for these models to incorporate, in a principled way, parameters that adequately address special characteristics of queries and documents. For example, the vector space model *assumes* that a query and a document are both represented by a term vector. However, the mapping from a query or a document to such a vector can be somehow arbitrary. Thus, because the model "sees" a document through its vector representation, there is no principled way to model the length of a document. As a result, heuristic parameters must be used (see, e.g., the pivot length normalization method (Singhal et al., 1996)). Similarly, in the BM25 retrieval formula, there is no direct modeling of queries, making it necessary to introduce heuristic parameters to incorporate query term frequencies (Robertson et al., 1995).

In order to be able to set parameters automatically, it is necessary to model queries and documents directly, and this is where the risk minimization retrieval framework has a

significant advantage over these traditional models, since it has the capability of modeling both queries and documents directly through statistical language models. Although a query and a document are similar in the sense that they are both text, they do have important differences. For example, queries are much shorter and often contain just a few keywords. Thus, from the viewpoint of language modeling, a query and a document require different language models. Practically, separating a query model from a document model has the important advantage of allowing *different* retrieval parameters for queries and documents when appropriate. In general, using statistical language models allows us to introduce all parameters through probabilistic models, and thus makes it possible to set the parameters automatically through statistical estimation methods.

In this chapter, we explore two-stage language models as a special case of the risk minimization framework, and show that we can achieve excellent retrieval performance through completely automatic parameter setting. In the rest of the chapter, we first derive a family of two-stage language models for retrieval using the risk minimization framework, and show that the specific two-stage smoothing method suggested in the previous chapter can be easily obtained as a special case. We then present methods for estimating the two parameters involved. Specifically, we propose a leave-one-out method for estimating the first-stage Dirichlet parameter and make use of a mixture model for estimating the second-stage interpolation parameter. Finally, we present experimental results on five different databases and four types of queries, and show that the two-stage smoothing method with the proposed parameter estimation method consistently gives retrieval performance that is close to, or better than, the best results of using a single smoothing method, achievable through an exhaustive parameter search.

The proposed two-stage smoothing method represents a step toward the goal of setting database-specific and query-specific retrieval parameters automatically, without the need for tedious experimentation. The effectiveness and robustness of the approach, along with the fact that there is no ad hoc parameter tuning involved, make it very useful as a solid baseline approach for the evaluation of retrieval models.

5.2 Derivation

In this section, we derive a general two-stage language model retrieval formula using the risk minimization retrieval framework presented in Chapter 3.

We start with the following risk ranking function with an independent loss function:

$$r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) = \int_{\Theta} l(\mathbf{d},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))p(\theta|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}})d\theta$$
$$= \int_{\Theta_Q}\int_{\Theta_D} l(\mathbf{d},\theta_Q,\theta_D,F(\mathcal{U}),F(\vec{\mathcal{S}}))p(\theta_Q \mid \mathbf{q},\mathcal{U})p(\theta_D \mid \mathbf{d},\vec{\mathcal{S}})\, d\theta_D\, d\theta_Q$$

Let us now consider the following special loss function, indexed by a small constant ϵ,

$$l_\epsilon(\mathbf{d},\theta_Q,\theta_D,F(\mathcal{U}),F(\vec{\mathcal{S}})) = \begin{cases} 0 & \text{if } \Delta(\theta_Q,\theta_D) \leq \epsilon \\ c & \text{otherwise} \end{cases}$$

where $\Delta : \Theta_Q \times \Theta_D \rightarrow \mathbb{R}$ is a model distance function, and c is a constant positive cost. Thus, the loss is zero when the query model and the document model are close to each other, and is c otherwise. Using this loss function, we obtain the following risk:

$$r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) = c - \int_{\Theta_D}\int_{\theta_Q \in S_\epsilon(\theta_D)} p(\theta_Q \mid \mathbf{q},\mathcal{U})\, p(\theta_D \mid \mathbf{d},\vec{\mathcal{S}})\, d\theta_Q\, d\theta_D$$

where $S_\epsilon(\theta_D)$ is the sphere of radius ϵ centered at θ_D in the parameter space.

Now, assuming that $p(\theta_D \mid \mathbf{d},\vec{\mathcal{S}})$ is concentrated on an estimated value $\hat{\theta}_D$, we can approximate the value of the integral over Θ_D by the integrand's value at $\hat{\theta}_D$. Note that the constant c can be ignored for the purpose of ranking. Thus, using $A \sim B$ to mean that A and B have the same effect for ranking, we have that

$$r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \overset{\text{rank}}{\approx} -\int_{\theta_Q \in S_\epsilon(\hat{\theta}_D)} p(\theta_Q \mid \mathbf{q},\mathcal{U})\, d\theta_Q$$

When θ_Q and θ_D belong to the same parameter space (i.e., $\Theta_Q = \Theta_D$) and ϵ is very small, the value of the integral can be approximated by the value of the function at $\hat{\theta}_D$ times a constant (the volume of $S_\epsilon(\hat{\theta}_D)$), and the constant can again be ignored for the purpose of ranking. That is,

$$r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \overset{\text{rank}}{\approx} -p(\hat{\theta}_D \mid \mathbf{q},\mathcal{U})$$

Therefore, using this risk we will be actually ranking documents according to $p(\hat{\theta}_D \mid \mathbf{q},\mathcal{U})$, i.e., the posterior probability that the user used the estimated document model as the query model. Applying Bayes' formula, we can rewrite this as

$$p(\hat{\theta}_D \mid \mathbf{q},\mathcal{U}) \propto p(\mathbf{q} \mid \hat{\theta}_D,\mathcal{U})p(\hat{\theta}_D \mid \mathcal{U}) \tag{5.1}$$

Equation 5.1 is our basic two-stage language model retrieval formula. Similar to the model discussed in (Berger and Lafferty, 1999), this formula has the following interpretation: $p(\mathbf{q} \mid \hat{\theta}_D, \mathcal{U})$ captures how well the estimated document model $\hat{\theta}_D$ explains the query, whereas $p(\hat{\theta}_D \mid \mathcal{U})$ encodes our prior belief that the user would use $\hat{\theta}_D$ as the query model. While this prior could be exploited to model different document sources or other document characteristics, in this chapter we assume a uniform prior.

The generic two-stage language model can be refined by specifying a concrete model $p(\mathbf{d} \mid \theta_D, \vec{\mathcal{S}})$ for generating documents and a concrete model $p(\mathbf{q} \mid \theta_Q, \mathcal{U})$ for generating queries; different specifications lead to different retrieval formulas. If the query generation model is the simplest unigram language model, we have the scoring procedure of the original language modeling approach proposed in (Ponte and Croft, 1998); that is, we first estimate a document language model and then compute the query likelihood using the estimated model. In the next section, we present the generative models that lead to the two-stage smoothing method suggested in the previous chapter.

5.3 The Two-Stage Smoothing Method

Let $\mathbf{d} = d_1 d_2 ... d_n$ denote a document, $\mathbf{q} = q_1 q_2 ... q_m$ denote a query, and $V = \{w_1, ..., w_{|V|}\}$ denote the words in the vocabulary. We consider the case where both θ_Q and θ_D are parameters of unigram language models, i.e., multinomial distributions over words in V.

The simplest generative model of a document is just the unigram language model θ_D, a multinomial. That is, a document would be generated by sampling words independently according to $p(\cdot \mid \theta_D)$, or

$$p(\mathbf{d} \mid \theta_D, \vec{\mathcal{S}}) = \prod_{i=1}^{n} p(d_i \mid \theta_D)$$

Each document is assumed to be generated from a potentially different model as assumed in the general risk minimization framework. Given a particular document $\mathbf{d}$, we want to estimate θ_D. We use a Dirichlet prior on θ_D with parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{|V|})$, given by

$$\text{Dir}(\theta \mid \alpha) \;=\; \frac{\Gamma(\sum_{i=1}^{|V|} \alpha_i)}{\prod_{i=1}^{|V|} \Gamma(\alpha_i)} \prod_{i=1}^{|V|} \theta_i^{\alpha_i - 1}$$

The parameters α_i are chosen to be $\alpha_i = \mu\, p(w_i \mid \vec{\mathcal{S}})$ where μ is a parameter and $p(\cdot \mid \vec{\mathcal{S}})$ is the "collection language model," which can be estimated based on a set of documents from

a source $\vec{\mathcal{S}}$. The posterior distribution of θ_D is given by

$$p(\theta_D \,|\, \mathbf{d}, \vec{\mathcal{S}}) \;\; \propto \;\; \prod_{w \in V} p(w \,|\, \theta_D)^{c(w,\mathbf{d}) + \mu p(w \,|\, \vec{\mathcal{S}}) - 1}$$

and so is also Dirichlet, with parameters $\alpha_i = c(w_i, \mathbf{d}) + \mu p(w_i \,|\, \vec{\mathcal{S}})$. Using the fact that the Dirichlet mean is $\alpha_j / \sum_k \alpha_k$, we have that

$$\begin{aligned} p_\mu(w \,|\, \hat{\theta}_D) &= \int_{\theta_D} p(w \,|\, \theta_D) p(\theta_D \,|\, \mathbf{d}, \vec{\mathcal{S}}) d\theta_D \\ &= \frac{c(w,d) + \mu \, p(w \,|\, \vec{\mathcal{S}})}{|d| + \mu} \end{aligned}$$

where $|\mathbf{d}| = \sum_{w \in V} c(w, \mathbf{d})$ is the length of $\mathbf{d}$. This is the Dirichlet prior smoothing method described in (Zhai and Lafferty, 2001b).

We now consider the query generation model. The simplest model is again the unigram language model θ_Q, which will result in a retrieval model with the Dirichlet prior as the single smoothing method. However, as observed in (Zhai and Lafferty, 2001b), such a model will not be able to explain the interactions between smoothing and the type of queries. In order to capture the common and non-discriminative words in a query, we assume that a query is actually composed of two types of words – "topic words" and "general English words" with possible overlaps. That is, a query is assumed to be generated by sampling words from a two-component mixture of multinomials, with one component being θ_Q and the other some query background language model $p(\cdot \,|\, \mathcal{U})$. That is,

$$p(\mathbf{q} \,|\, \theta_Q, \lambda, \mathcal{U}) = \prod_{i=1}^{m} ((1-\lambda) p(q_i \,|\, \theta_Q) + \lambda p(q_i \,|\, \mathcal{U}))$$

where λ is a parameter, roughly indicating the amount of "noise" in $\mathbf{q}$.

Combining our estimate of θ_D with this query model, we have the following retrieval scoring formula for document $\mathbf{d}$ and query $\mathbf{q}$.

$$\begin{aligned} & p(\mathbf{q} \,|\, \hat{\theta}_D, \lambda, \mathcal{U}) \;\; = \\ &= \;\; \prod_{i=1}^{m} \Big((1-\lambda) p(q_i \,|\, \hat{\theta}_D) + \lambda \, p(q_i \,|\, \mathcal{U}) \Big) \\ &= \;\; \prod_{i=1}^{m} \left((1-\lambda) \frac{c(q_i, d) + \mu \, p(q_i \,|\, \vec{\mathcal{S}})}{|\mathbf{d}| + \mu} + \lambda \, p(q_i \,|\, \mathcal{U}) \right) \end{aligned}$$

In this formula, the document language model is effectively smoothed in two steps. First, it is smoothed with a Dirichlet prior, and second, it is interpolated with a query background

model. This is precisely the two-stage smoothing method suggested in the previous chapter, where the goal is to decouple the dual role of smoothing.

The query background model $p(\cdot \mid \mathcal{U})$ is in general different from the collection language model $p(\cdot \mid \vec{\mathcal{S}})$, and can be estimated using query logs of the current user $\mathcal{U}$ or similar users. With insufficient data to estimate $p(\cdot \mid \mathcal{U})$, however, we can assume that $p(\cdot \mid \vec{\mathcal{S}})$ would be a reasonable approximation of $p(\cdot \mid \mathcal{U})$. In this form, the two-stage smoothing method is essentially a combination of Dirichlet prior smoothing with Jelinek-Mercer smoothing (Zhai and Lafferty, 2001b). Indeed, it is very easy to verify that when $\lambda = 0$, we end up with just the Dirichlet prior smoothing, whereas when $\mu = 0$, we will have Jelinek-Mercer smoothing. Since the combined smoothing formula still follows the general smoothing scheme discussed in (Zhai and Lafferty, 2001b), it can be implemented very efficiently.

In the next section, we empirically explore the parameter space of the two-stage smoothing method, and examine whether the method is effective in decoupling the estimation role and the query modeling role of smoothing.

5.4 Empirical Exploration of Two-Stage Smoothing

In the experiments reported in the previous chapter, we observed strong interactions between the type of queries and the sensitivity of performance to smoothing. The idea of the two-stage smoothing method is to factor out the influence of the query on smoothing through a second-stage interpolation smoothing. To see whether we can expect to achieve this goal empirically, we explore the parameter space of the two-stage smoothing method, and examine the sensitivity of retrieval performance to the two smoothing parameters. As we will see, the two-stage smoothing method does help reveal more consistent sensitivity patterns through decoupling the two roles.

The experimental setup is the same as in the previous chapter, Section 4.4, but we only used the three largest collections.

Figure 5.1 shows how two-stage smoothing reveals a more regular sensitivity pattern than the single-stage smoothing methods studied earlier in this paper. The three figures show how the precision varies when we change the prior value in Dirichlet smoothing on three different testing collections respectively (Trec7 ad hoc task, Trec8 ad hoc task, and

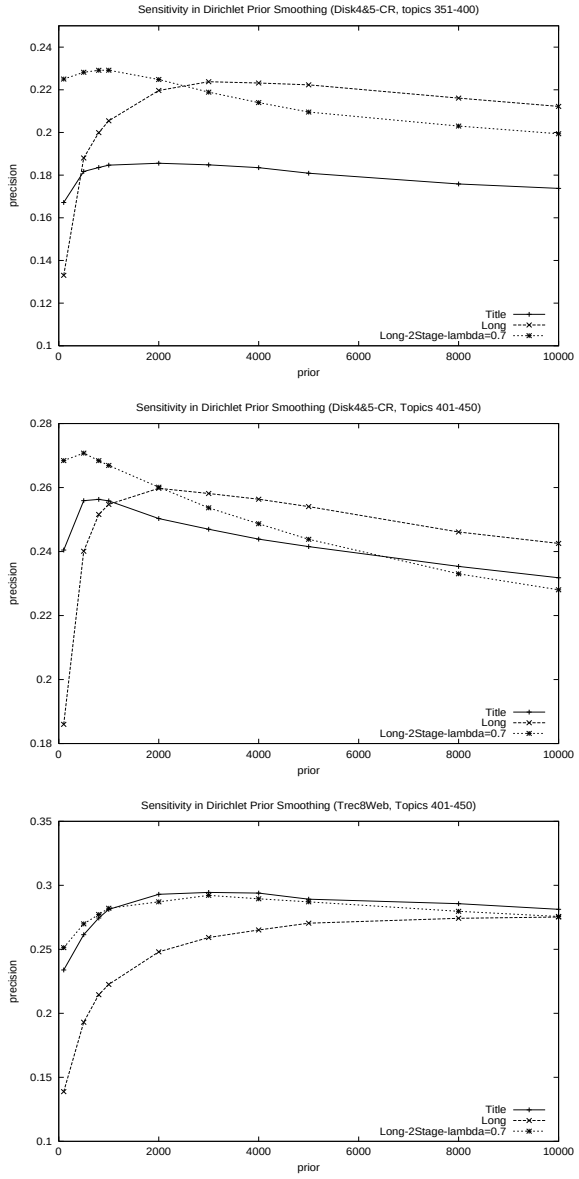

Figure 5.1: Two-stage smoothing reveals a more consistent sensitivity pattern of Dirichlet prior for both title queries and long queries. Each figure is for a different collection: Trec7 (top), Trec8 (middle), and web testing collection (bottom).

Trec8 web task). [1] The three lines in each figure correspond to (1) Dirichlet smoothing with title queries; (2) Dirichlet smoothing with long queries; and (3) Two-stage smoothing

[1] These are exactly the same as trec7T, trec7L, trec8T, trec8L, web8T, and web8L described in Section 4.4, but are labeled differently here in order to clarify the effect of two-stage smoothing.

(Dirichlet + Jelinek-Mercer, $\lambda = 0.7$) with long queries. [2] The first two are thus based on a single smoothing method (Dirichlet prior), while the third uses two-stage smoothing with a near optimal λ.

In each of the figure, we see that

1. If Dirichlet smoothing is used alone, the sensitivity pattern for long queries is very different from that for title queries; the optimal setting of the prior is clearly different for title queries than for long queries.

2. In two-stage smoothing, with the help of Jelinek-Mercer smoothing in the second role, the sensitivity pattern of Dirichlet smoothing is much more similar for title queries and long queries, i.e., it becomes much less sensitive to the type/length of queries. By factoring out the second role of smoothing, we see a more consistent and stable behavior for Dirichlet smoothing, which is filling the first role. Indeed, when comparing the Trec7 figure (top) and the Trec8 (middle), which involve different topic sets but the same document collection, we can see that the optimal value for the prior parameter on both figures is generally between 500 and 2000, despite the difference in queries.

3. Given a topic set, the *same* λ value can stabilize the sensitivity pattern of Dirichlet smoothing on *different* collections. The Trec8 figure (middle) and web figure (bottom) involve the same topic set but different collections. In comparing them, note that the same λ value (0.7) can cause the Dirichlet prior to behave similarly for both title queries and long queries on each of the two collections.

4. The optimal value of the prior parameter in the Dirichlet prior method is generally sensitive to the collection. This can be seen by comparing the Trec8 figure (middle) with the web figure (bottom) and noticing that the position of the optimal prior is different.

It should also be noted in these figures that with the help from the Jelinek-Mercer method in two-stage smoothing, Dirichlet smoothing can achieve a higher precision on long queries than is attainable with single-stage smoothing alone. Of course, this may be because Jelinek-Mercer is just a better method for long queries, but the two-stage results are often also better than the best result of using Jelinek-Mercer alone (see Table 5.4).

[2] 0.7 is a near optimal value of λ when the Jelinek-Mercer method is applied alone.

It is worth mentioning that on the web data (Figure 5.1, bottom), the title queries perform much better than the long queries with Dirichlet prior smoothing alone. But with two-stage smoothing, the long queries perform similarly to the title queries, though still slightly worse. This is in contrast to the clear improvement of long queries over title queries in the Trec8 collection (Figure 5.1, middle). The topic set is exactly the same for both figures, so this can only be explained by the difference in the collections. One possibility is that the extra words introduced in the long queries are not so useful for retrieval on the web database as on the Trec8 database. For example, it may be that the extra words are more ambiguous on the web database, but are used with only the "right" sense on the Trec8 database. More analysis is needed to fully understand this.

Collection	Best Jelinek-Mercer	Best Dirichlet	(near optimal) Two-Stage
TREC7L	0.222(λ=0.8)	0.224 (μ=3,000)	0.229(λ=0.7, μ=800)
TREC8L	0.265(λ=0.8)	0.260(μ=2,000)	0.268(λ=0.7, μ=800)
WEB8L	0.259(λ=0.5)	0.275(μ=10,000)	0.292(λ=0.7, μ=3,000)
TREC7T	0.167(λ=0.3)	0.186 (μ=2,000)	0.184(λ=0, μ=800)
TREC8T	0.239(λ=0.2)	0.256(μ=800)	0.256(λ=0, μ=800)
WEB8T	0.243(λ=0.01)	0.294(μ=3,000)	0.294(λ=0, μ=3,000)

Table 5.1: Comparison of the average precision between near optimal two-stage smoothing and the best one-stage smoothing. Setting $\lambda = 0.7$ for long queries, and $\lambda = 0$ for title queries, and setting $\mu = 800$ for Trec7/Trec8 database, and $\mu = 3,000$ for the web database, the two-stage smoothing performs as well as the *best* single-stage smoothing method, often better.

On the surface, we now have two parameters, rather than one, to worry about. However, as these results show, the two parameters λ and μ can be interpreted in a meaningful way. λ is intended to be a query-related parameter, and should be smaller if the query is more verbose and long. It can be roughly interpreted as modeling the expected "noise" in the query. μ is a document-related parameter, which controls the amount of probability mass assigned to unseen words. The optimal value of μ can be expected to be relatively stable for a fixed document collection, while the optimal setting of Jelinek-Mercer λ can be expected to be more correlated with the query. This makes it possible to optimize μ based on only the document collection without depending on queries and similarly optimize λ primarily based on queries. In the next section, we present methods for estimating μ and λ from data.

5.5 Parameter Estimation for Two-Stage Smoothing

5.5.1 Estimating μ

The purpose of Dirichlet prior smoothing in the first stage is to address the estimation bias due to the fact that a document is an extremely small amount of data with which to estimate a unigram language model. More specifically, it is to discount the maximum likelihood estimate appropriately and assign non-zero probabilities to words not observed in a document; this is the usual role of language model smoothing. A useful objective function for estimating smoothing parameters is the "leave-one-out" likelihood, that is, the sum of the log-likelihoods of each word in the observed data computed in terms of a model constructed based on the data with the target word excluded ("left out"). This criterion is essentially based on cross-validation, and has been used to derive several well-known smoothing methods including the Good-Turing method (Ney et al., 1995).

Formally, let $\mathcal{C} = \{d_1, d_2, ..., d_N\}$ be the collection of documents. Using our Dirichlet smoothing formula, the leave-one-out log-likelihood can be written as

$$\ell_{-1}(\mu \,|\, \mathcal{C}) \;=\; \sum_{i=1}^{N} \sum_{w \in V} c(w, d_i) \log \left(\frac{c(w, d_i) - 1 + \mu p(w|\mathcal{C})}{|d_i| - 1 + \mu} \right)$$

Thus, our estimate of μ is

$$\hat{\mu} = \arg\max_{\mu} \ell_{-1}(\mu|\mathcal{C})$$

which can be easily computed using Newton's method. The update formula is

$$\mu^{(k+1)} = \mu^{(k)} - g(\mu^{(k)}) / g'(\mu^{(k)})$$

where the first and second derivatives of ℓ_{-1} are given by

$$g(\mu) \;=\; \ell'_{-1}(\mu) \;=\; \sum_{i=1}^{N} \sum_{w \in V} \frac{c(w, d_i)((|d_i| - 1)p(w \,|\, \mathcal{C}) - c(w, d_i) + 1)}{(|d_i| - 1 + \mu)(c(w, d_i) - 1 + \mu p(w \,|\, \mathcal{C}))}$$

and

$$g'(\mu) \;=\; \ell''_{-1}(\mu) \;=\; -\sum_{i=1}^{N} \sum_{w \in V} \frac{c(w, d_i)((|d_i| - 1)p(w \,|\, \mathcal{C}) - c(w, d_i) + 1)^2}{(|d_i| - 1 + \mu)^2 (c(w, d_i) - 1 + \mu p(w \,|\, \mathcal{C}))^2}$$

Collection	avg. doc length	max. doc length	vocab. size	$\hat{\mu}$
AP88-89	446	2678	254872	640.643
WSJ87-92	435	8980	260259	792.001
ZF1-2	455	53753	447066	719.637

Table 5.2: Estimated values of μ along with database characteristics.

Since $g' \leq 0$, as long as $g' \neq 0$, the solution will be a global maximum. In our experiments, starting from value 1.0 the algorithm always converges. Each iteration goes through all the terms in all the documents, so the complexity for each iteration is $O(NL)$, where N is the total number of documents in the collection and L is the average number of distinct terms in a document. This is not a big concern, since the estimation of μ can be done at the indexing time, independent of queries.

The estimated values of μ for three databases are shown in Table 5.2. There is no clear correlation between the database characteristics shown in the table and the estimated value of μ.

5.5.2 Estimating λ

With the query model hidden, the query likelihood is

$$p(\mathbf{q} \,|\, \lambda, \mathcal{U}) = \int_{\Theta_Q} \prod_{i=1}^{m} ((1-\lambda) p(q_i \,|\, \theta_Q) + \lambda p(q_i \,|\, \mathcal{U}))\, p(\theta_Q \,|\, \mathcal{U})\, d\theta_Q$$

In order to estimate λ, we approximate the query model space by the set of all N estimated document language models in our collection. That is, we will approximate the integral with a sum over all the possible document language models estimated on the collection, or

$$p(\mathbf{q} \,|\, \lambda, \mathcal{U}) = \sum_{i=1}^{N} \pi_i \prod_{j=1}^{m} ((1-\lambda) p(q_j \,|\, \hat{\theta}_{d_i}) + \lambda p(q_j \,|\, \mathcal{U}))$$

where $\pi_i = p(\hat{\theta}_{d_i} \,|\, \mathcal{U})$, and $p(\cdot \,|\, \hat{\theta}_{d_i})$ is the smoothed unigram language model estimated based on document d_i using the Dirichlet prior approach.

Thus, we assume that the query is generated from a mixture of N document models with unknown mixing weights $\{\pi_i\}_{i=1}^{N}$. With this setup, the parameters λ and $\{\pi_i\}_{i=1}^{N}$ can

be estimated using the EM algorithm. The update formulas are

$$\pi_i^{(k+1)} = \frac{\pi_i^{(k)} \prod_{j=1}^{m}((1-\lambda^{(k)})\, p(q_j \,|\, \hat{\theta}_{d_i}) + \lambda^{(k)} p(q_j \,|\, \mathcal{U}))}{\sum_{i'=1}^{N} \pi_{i'}^{(k)} \prod_{j=1}^{m}((1-\lambda^{(k)}) p(q_j \,|\, \hat{\theta}_{d_{i'}}) + \lambda^{(k)} p(q_j \,|\, \mathcal{U}))}$$

and

$$\lambda^{(k+1)} = \frac{1}{m} \sum_{i=1}^{N} \pi_i^{(k+1)} \sum_{j=1}^{m} \frac{\lambda^{(k)} p(q_j \,|\, \mathcal{U})}{(1-\lambda^{(k)}) p(q_j \,|\, \hat{\theta}_{d_i}) + \lambda^{(k)} p(q_j \,|\, \mathcal{U})}$$

Note that leaving $\{\pi_i\}_{i=1}^N$ free is important, because what we really want is not to maximize the likelihood of generating the query from *every* document in the collection. Instead, we want to find a λ that can maximize the likelihood of the query given *relevant* documents. With $\{\pi_i\}_{i=1}^N$ free to estimate, we would indeed allocate higher weights to documents that predict the query well in our likelihood function; presumably, these documents are also more likely to be relevant. Unfortunately, if we intend to find the exact maximum likelihood estimate of all $\{\pi_i\}_{i=1}^N$, we will end up assigning the entire probability mass to one *single* document. This is because, for any given λ, there always exists a smoothed document model that gives the highest likelihood for the query, thus setting $\pi_i = 1$ for this document always gives a higher overall likelihood than otherwise. Clearly this is not desirable and empirically it leads to non-optimal performance. To solve this problem, we initialize $\{\pi_i\}_{i=1}^N$ with a *uniform* distribution and λ with 0.5, and allow early stop in the EM algorithm (after 10 iterations in our experiments). This strategy allows us to obtain a relatively smooth estimate of $\{\pi_i\}_{i=1}^N$ and works well empirically. While the number of EM iterations to carry out becomes another parameter that needs to be set, the performance is not very sensitive to the exact number of iterations, as long as it is not too large (e.g., smaller than 20); actually, it is often "safer" to use a smaller number of iterations, in the sense that the performance would not be too much different from the optimal performance.

Each EM iteration involves the computation of the query likelihood for each document, so the complexity is about the same as scoring all the documents. With the inverted index, the complexity for each iteration is $O(M+N)$, where M is the total number of inverted index entries under all query terms, and N is the total number of documents in the collection. Since we have to estimate a different λ for each query, the computational complexity is a concern in practice. One possible way to improve the efficiency is to use only a small number of documents of the highest π_i's, since the estimated λ value is mainly determined

by these documents. How to develop more efficient methods for estimating λ would be a topic for further research.

5.6 Effectiveness of the Parameter Estimation Methods

To evaluate the two parameter estimation methods for the two-stage smoothing method, we tested it on the same 12 testing collections as we used for studying the query factors (see Section 4.8). These collections represent a good diversity in the types of queries and documents. However, they are all homogeneous databases and are relatively small. In order to further test the robustness of the two-stage smoothing method, we also tested it on three other much bigger and more heterogeneous TREC collections. These are the official ad hoc retrieval collections from TREC-7, TREC-8, and the TREC-8 small web track, and are the same as what we use in the experiments described in Section 4.4. [3] Since these topics do not have a concept field, we have only three types of queries: short-keyword, short-verbose, and long-verbose. Again, we perform minimum pre-processing – only a Porter stemmer is used, and no stop words are removed.

For each testing collection, we compare the retrieval performance of the estimated two-stage smoothing parameters with the best results achievable using a single smoothing method. The best results of a single smoothing method are obtained through an exhaustive search on its parameter space, so they are the ideal performance of the smoothing method. In all our experiments, we used the collection language model to approximate the query background model.

The results are shown in Table 5.3 and Table 5.4 for the small collections and large collections, respectively. The four types of queries are abbreviated with the two initial letters (e.g., SK for Short-Keyword). The standard TREC evaluation procedure for ad hoc retrieval is followed, and we have considered four performance measures – non-interpolated average precision, initial precision (i.e., precision at 0.0 recall), and precision at 10 and 20 documents. In all the results, we see that the performance of two-stage smoothing with the estimated parameter values is consistently very close to or better than the best performance of a single method by all three measures. Only in a few cases, is the difference statistically significant (indicated with an asterisk).

To quantify the sensitivity of the retrieval performance to the smoothing parameter for

[3] We label these collections differently here to make the labeling consistent with that in Table 5.3.

Collection	Query	Method	Avg. Prec.	(Median)	Init. Prec.	Prec@10	Prec@20
AP88-89	SK	Best JM	0.203	(0.194)	0.573	0.310	0.283
		Best Dir	**0.230**	(0.224)	**0.623**	0.356	**0.332**
		Two-Stage	0.222*		0.611	**0.358**	0.317
	LK	Best JM	0.368	(0.362)	**0.767**	**0.509**	0.469
		Best Dir	**0.376**	(0.368)	0.755	0.506	0.475
		Two-Stage	0.374		0.754	0.505	**0.480**
	SV	Best JM	0.188	(0.158)	0.569	0.309	0.272
		Best Dir	**0.209**	(0.195)	**0.609**	0.338	0.304
		Two-Stage	0.204		0.598	**0.339**	**0.305**
	LV	Best JM	0.288	(0.263)	**0.711**	0.430	0.391
		Best Dir	**0.298**	(0.285)	0.704	**0.453**	**0.403**
		Two-Stage	0.292		0.689	0.444	0.400
WSJ87-92	SK	Best JM	0.194	(0.188)	0.629	0.364	0.330
		Best Dir	**0.223**	(0.218)	0.660	**0.412**	**0.376**
		Two-Stage	0.218*		**0.662**	0.409	0.366
	LK	Best JM	0.348	(0.341)	0.814	**0.575**	**o.524**
		Best Dir	0.353	(0.343)	0.834	0.562	0.507
		Two-Stage	**0.358**		**0.850***	0.572	0.523
	SV	Best JM	0.172	(0.158)	0.615	0.346	0.314
		Best Dir	0.196	(0.188)	0.638	0.389	0.333
		Two-Stage	**0.199**		**0.660**	**0.391**	**0.344**
	LV	Best JM	0.277	(0.252)	**0.768**	0.481	**0.452**
		Best Dir	0.282	(0.270)	0.750	0.480	0.442
		Two-Stage	**0.288***		0.762	**0.497**	0.449
ZF1-2	SK	Best JM	0.179	(0.170)	0.455	0.220	0.193
		Best Dir	**0.215**	(0.210)	**0.514**	**0.265**	0.226
		Two-Stage	0.200		0.490	0.256	**0.227**
	LK	Best JM	**0.306**	(0.290)	0.675	0.345	0.300
		Best Dir	**0.326**	(0.316)	0.681	**0.376**	**0.322**
		Two-Stage	0.322		**0.696**	0.368	**0.322**
	SV	Best JM	0.156	(0.139)	0.450	0.208	0.174
		Best Dir	**0.185**	(0.170)	0.456	0.225	0.185
		Two-Stage	0.181		**0.487**	**0.246***	0.203
	LV	Best JM	0.267	(0.242)	0.593	0.300	0.258
		Best Dir	**0.279**	(0.273)	0.606	0.329	0.272
		Two-Stage	**0.279***		**0.618**	**0.334**	**0.278**

Table 5.3: Comparison of the estimated two-stage smoothing with the best single stage smoothing methods on small collections. The best number for each measure is shown in boldface. An asterisk (*) indicates that the difference between the two-stage smoothing performance and the best one-stage smoothing performance is statistically significant according to the Wilcoxin signed rank test at the level of 0.05.

single smoothing methods, we also show (in parentheses) the median average precision for all the parameter values that are tried. [4] We see that for Jelinek-Mercer the sensitivity is clearly higher on verbose queries than on keyword queries; the median is usually much

[4]For Jelinek-Mercer, we tried 13 values $\{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$; for Dirichlet prior, we tried 10 values $\{100, 500, 800, 1000, 2000, 3000, 4000, 5000, 8000, 10000\}$.

Collection	Query	Method	Avg. Prec.	(Median)	Init. Prec.	Prec@10	Prec@20
Disk4&5-CR	Trec7-SK	Best JM	0.167	(0.165)	0.632	0.366	0.315
		Best Dir	**0.186**	(0.182)	**0.688**	0.412	0.342
		Two-Stage	0.182		0.673	**0.420**	**0.357**
	Trec7-SV	Best JM	0.173	(0.138)	0.646	0.392	0.342
		Best Dir	**0.182**	(0.168)	**0.656**	**0.416**	0.340
		Two-Stage	0.181		0.655	**0.416**	**0.348**
	Trec7-LV	Best JM	0.222	(0.195)	0.723	**0.476**	0.401
		Best Dir	0.224	(0.212)	**0.763**	0.456	0.383
		Two-Stage	**0.230**		0.760	0.466	**0.412**
	Trec8-SK	Best JM	0.239	(0.237)	0.621	0.438	0.378
		Best Dir	0.256	(0.244)	0.717	**0.448**	0.398
		Two-Stage	**0.257**		**0.719**	**0.448**	**0.405**
	Trec8-SV	Best JM	**0.231**	(0.192)	0.687	0.416	**0.357**
		Best Dir	0.228	(0.222)	0.670	0.400	0.337
		Two-Stage	**0.231**		**0.719**	**0.440**	0.354
	Trec8-LV	Best JM	0.265	(0.234)	**0.789**	**0.504**	**0.434**
		Best Dir	0.260	(0.252)	0.753	0.484	0.400
		Two-Stage	**0.268**		0.787	0.478	0.400
Web	Trec8-SK	Best JM	0.243	(0.212)	0.607	0.348	0.293
		Best Dir	**0.294**	(0.281)	**0.756**	**0.448**	**0.374**
		Two-Stage	0.278*		0.730	0.426	0.358
	Trec8-SV	Best JM	0.203	(0.191)	0.611	0.340	0.284
		Best Dir	**0.267**	(0.249)	**0.699**	**0.408**	**0.325**
		Two-Stage	0.253		0.680	0.398	0.330
	Trec8-LV	Best JM	0.259	(0.243)	**0.790**	0.422	0.348
		Best Dir	0.275	(0.248)	0.752	0.410	0.343
		Two-Stage	**0.284**		0.781	**0.442**	**0.362**

Table 5.4: Comparison of the estimated two-stage smoothing with the best single stage smoothing methods on large collections. The best number for each measure is shown in boldface. An asterisk (*) indicates that the difference between the two-stage smoothing performance and the best one-stage smoothing performance is statistically significant according to the Wilcoxin signed rank test at the level of 0.05.

lower than the best performance for verbose queries. This means that it is much harder to tune the λ in Jelinek-Mercer for verbose queries than for keyword queries. Interestingly, for Dirichlet prior, the median is often just slightly below the best, even when the queries are verbose. (The worst cases are significantly lower though.) From the sensitivity curves in the three figures (Figure 4.7, Figure 4.8, and Figure 4.9) presented in the previous chapter, we see that as long as we set a relatively large value for μ in the Dirichlet prior, the performance will not be much worse than the best performance, and the median is most likely at a large value for μ. This immediately suggests that we can expect to perform reasonably well if we simply set μ to some "safe" large value. However, it is clear from the results in Table 5.3 and Table 5.4, that such a simple approach would not perform so well as our parameter estimation methods. Indeed, the two-stage performance is always better than the median,

except for three cases of short-keyword queries when it is slightly worse. Since the Dirichlet prior smoothing dominates the two-stage smoothing effect for these short-keyword queries (due to "little noise"), this somehow suggests that the leave-one-out method might have underestimated μ.

Note that, in general, Jelinek-Mercer has not performed as well as Dirichlet prior in all our experiments. But in two cases of verbose queries (Trec8-SV and Trec8-LV on the Trec7/8 database), it does outperform Dirichlet prior. In these two cases, the two-stage smoothing method performs either as well as or better than the Jelinek-Mercer. Thus, the two-stage smoothing performance appears to always track the *best performing* single method at its optimal parameter setting.

The performance of two-stage smoothing does not reflect the performance of a "full-fledged" language modeling approach, which would involve more sophisticated feedback models (Lafferty and Zhai, 2001a; Lavrenko and Croft, 2001; Zhai and Lafferty, 2001a). Thus, it is really not comparable with the performance of other TREC systems. Yet some of the performance figures shown here are actually competitive when compared with the performance of the official TREC submissions (e.g., the performance on the TREC-8 ad hoc task and the TREC-8 web track).

These results of the two-stage smoothing method are very encouraging, especially because the approach involves no ad hoc parameter tuning in the retrieval process. Both μ and λ are automatically estimated based on a specific database and query; μ is completely determined by the given database, and λ is determined by the database and the query together. The method appears to be quite robust according to our experiments with all the different types of queries and different databases.

5.7 Conclusions and Further Work

In this chapter we derive a family of two-stage language models using the risk minimization retrieval framework, and present a two-stage smoothing method as a special case. The two-stage language modeling approach generalizes the original query likelihood method proposed in (Ponte and Croft, 1998) by incorporating a query generative model, making it possible to explicitly capture the different influences of the query and document collection on the optimal setting of smoothing parameters. An important advantage of the two-stage language models is that they explicitly capture the different influences of the query and document collection on smoothing, making it easier to estimate smoothing parameters in-

dependently according to different documents and different queries.

As a special case of the two-stage language modeling approach, we formally derive the two-stage smoothing method suggested in the previous chapter, where the method is empirically motivated by the observation of the dual role of smoothing. In the first stage of this two-stage smoothing method, the document language model is smoothed using a Dirichlet prior with the collection language model as the reference model; in the second stage, the smoothed document language model is further interpolated with a query background language model. The two-stage smoothing method is shown to reveal a more regular sensitivity pattern of smoothing empirically.

We propose a leave-one-out method for estimating the first-stage Dirichlet prior parameter and a mixture model for estimating the second-stage interpolation parameter. These methods allow us to set the retrieval parameters automatically, yet adaptively, according to different databases and queries. Evaluation on five different databases and four types of queries indicates that the two-stage smoothing method with the proposed parameter estimation scheme consistently gives retrieval performance that is close to, or better than, the best results attainable using a one-stage smoothing method, achievable only through an exhaustive parameter search. The effectiveness and robustness of the two-stage smoothing approach, along with the fact that no ad hoc parameter tuning is involved, make it a solid baseline approach for evaluating retrieval models.

Ideally, one should tie the setting of parameters directly to the optimization of performance. Unfortunately, this is difficult, if not impossible, for ad hoc text retrieval due to the fact that the performance can only be measured subjectively by a user. Of course, when relevance feedback is available, machine learning techniques can be exploited to set parameters by optimizing the performance on documents with known judgments. Since we are mainly considering the ad hoc retrieval scenario, the parameter setting problem is, in some sense, inherently empirical. In this sense, the maximum likelihood parameter estimation may not necessarily guarantee to optimize the retrieval performance, even though it has certain guarantees on the optimality of the estimated language model. However, our methods still have a significant advantage over traditional parameter tuning methods, since it is possible to see how we should improve the language models and the estimation methods through the analysis of such non-optimality when it happens, whereas in traditional parameter tuning, one rarely has any guidance on how to further improve the model. In any case, the consistent near-optimal performance demonstrated by the parameter estimation methods in our extensive evaluation suggests that our methods for automatic parameter setting

are robust and effective.

There are several interesting further research directions. First, while we have shown that automatic two-stage smoothing gives retrieval performance close to the best results attainable using a one-stage smoothing method, we have not yet analyzed the optimality of the estimated parameter values in the two-stage parameter space. For example, it would be important to see the relative optimality of the estimated μ and λ when fixing one of them. Second, it would also be interesting to explore other estimation methods. For example, μ might be regarded as a hyperparameter in a hierarchical Bayesian approach. For the estimation of the query model parameter λ, it would be interesting to explore more efficient methods than the mixture model that we have used. It would also be interesting to try different query background models. One possibility is to estimate the background model based on resources such as past queries, in addition to the collection of documents. Finally, it is possible to exploit the query background model to address the issue of redundancy in the retrieval results. Specifically, a biased query background model may be used to represent/explain the sub-topics that a user has already encountered (e.g., through reading previously retrieved results), in order to focus ranking on the *new* sub-topics in a relevant set of documents. This direction will be further explored in Chapter 7.

Chapter 6

KL-divergence Retrieval Models

In this chapter, we study another special case of the risk minimization retrieval framework, which not only generalizes the existing language modeling approach, but also allows us to deal with feedback in a more natural way.

We derive a family of unigram retrieval models using a KL-divergence loss function in the risk minimization framework. In effect, these models all score documents by computing the KL-divergence of the query model and document model. The KL-divergence retrieval approach extends the the query likelihood approach by incorporating a query language model. The lack of a query model in previous work on the language modeling approach has made it unnatural to incorporate feedback. The KL-divergence approach involves estimating a query language model, in addition to the document language model, making it possible to treat feedback naturally as query model updating. We propose two specific query model updating algorithms based on feedback documents. Evaluation indicates that both algorithms are effective for feedback. Thus we show that we can exploit feedback documents to improve our estimation of the query model, and the improved query model results in better retrieval performance.

6.1 Derivation

In this section, we derive the KL-divergence retrieval models formally using the risk minimization framework. We start with the general probabilistic similarity model derived in Chapter 3, which we repeat here

$$r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \propto \Delta(\widehat{\theta}_D,\widehat{\theta}_Q)$$

As a special case of this model, we assume that θ_Q and θ_D are the parameters of unigram language models (called the query topic language model and the document topic language model respectively), and choose as the distance function the Kullback-Leibler divergence.

Given two probability mass functions $p(x)$ and $q(x)$, $D(p||q)$, the Kullback-Leibler divergence (or relative entropy) between p and q is defined as

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

It is easy to show that $D(p||q)$ is always non-negative and is zero if and only if $p = q$. Even though it is not a true distance between distributions (because it is not symmetric and does not satisfy the triangle inequality), it is still often useful to think of the KL-divergence as a "distance" between distributions (Cover and Thomas, 1991).

Apply the KL-divergence function to θ_Q and θ_D, we see that

$$\Delta(\theta_Q,\theta_D) = D(\theta_Q||\theta_D) = \sum_w p(w \mid \theta_Q) \log \frac{p(w \mid \theta_Q)}{p(w \mid \theta_D)}$$

And,

$$\begin{aligned} r(\mathbf{d}|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) &\propto -\sum_w p(w \mid \widehat{\theta}_{\mathbf{q}}) \log p(w \mid \widehat{\theta}_{\mathbf{d}}) + \mathit{cons}(\mathbf{q}) \\ &\propto -\sum_w p(w \mid \widehat{\theta}_{\mathbf{q}}) \log p(w \mid \widehat{\theta}_{\mathbf{d}}) \end{aligned}$$

Thus the ranking function is essentially the cross entropy of the query language model with respect to the document language model. The dropped constant is the query model entropy (with a different sign). The value of the cross entropy is always larger than or equal to the query model entropy. The minimum value (i.e., query model entropy) is achieved when $\widehat{\theta}_d$ is identical to $\widehat{\theta}_q$, which makes sense for retrieval.

The KL-divergence model covers the popular query likelihood ranking function as a special case. Indeed, suppose $\widehat{\theta}_{\mathbf{q}}$ is just the empirical distribution of the query $\mathbf{q} = (q_1, q_2, \ldots, q_m)$, That is, $\widehat{\theta}_{\mathbf{q}}$ is the language model given by

$$p(w \mid \widehat{\theta}_{\mathbf{q}}) = \frac{1}{m} \sum_{i=1}^{m} \delta(w, q_i)$$

where $\delta(w, q_i)$ is the indicator function. We obtain

$$r(\mathbf{d}|\mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) \quad \propto \quad -\frac{1}{m}\sum_{i=1}^{m} \log p(q_i \mid \widehat{\theta}_{\mathbf{d}})$$

This is precisely the log-likelihood criterion used by Ponte and Croft (1998) in introducing the language modeling approach, and that has been used in all work on the language modeling approach to date. Later in this chapter, we will develop new methods to estimate a model $\widehat{\theta}_Q$, and demonstrate that the new models perform significantly better than using the empirical distribution as $\widehat{\theta}_Q$.

6.2 Interpretation of the Model

The KL-divergence retrieval model is a special case of the general probabilistic distance model. Interestingly, the general probabilistic distance model (thus also the KL-divergence model) has an intuitive interpretation that is very similar to the vector space model. Recall that the vector space model has three basic components: (1) a query term vector; (2) a document term vector; and (3) a vector similarity measure (e.g., cosine). The probabilistic distance model also has three similar components: (1) a query language model; (2) a document language model; and (3) a model-similarity measure (e.g., the KL-divergence). However, there is also one very important difference – the representation. In the vector space model, a document (or a query) is represented by a heuristically weighted term vector, while in the probabilistic distance model, it is represented by a unigram language model. The heuristic term weighting used in the vector space model makes the whole model mostly empirical, whereas the use of a unigram language model as representation makes it possible to exploit the principles of statistical parameter estimation to improve a representation for the purpose of retrieval.

The KL-divergence model also appears to be similar to a probability distribution model proposed in (Wong and Yao, 1989) that uses the information-theoretic retrieval strategy. However, the KL-divergence model is actually much more general and flexible because of its explicit modeling of the query and documents. In (Wong and Yao, 1989), the multinomial term distribution was primarily proposed as an *alternative representation* of documents and query (in the vector space model sense), *not a generative model* for documents or query. Thus, it is not surprising that the issue of model estimation has not been considered at all and the term distribution representation is naturally assumed to be best approximated

by the relative frequency of terms. The limitation in modeling can be seen more clearly from the claimed difficulty with using the KL-divergence function directly as the similarity measure, since model smoothing has not been considered as a possibility (Wong and Yao, 1989).

Note that the KL-divergence distance function is only one of the many possibilities of measuring model similarity. [1] However, in this thesis, we will not explore other distance functions, but will focus on the derived unigram KL-divergence model. The model suggests that documents can be ranked by the KL-divergence function value between the estimated query unigram language model and the estimated document unigram language model. In the following sections, we will discuss the estimation of both the query model and the document model.

6.3 Estimation of Document Language Model (θ_D) and Query Language Model (θ_Q)

In the KL-divergence retrieval model, the retrieval problem boils down to the problem of estimating a document language model θ_D and a query language model θ_Q. Depending on the actual language models assumed, there are many possibilities here. For example, for document model estimation, different smoothing methods lead to different estimated models. This has already been discussed in Chapter 4. Since ranking based on KL-divergence is equivalent to ranking based on cross entropy, it can be shown that, for a given query model, the computation of the KL-divergence retrieval formula is as efficient as that of the query likelihood formula, which is as efficient as the traditional TF-IDF retrieval formula for most simple smoothing methods, as we have shown in Chapter 4. Of course, there are many other ways to estimate document models. In particular, mixture models can be used to reduce the noise in documents, resulting in a document distillation method, which achieves an effect like TF-IDF weighting of document terms, and is similar to the query distillation approach to be discussed below.

However, here we focus our study on the estimation of the query model, which is very much related to the issue of feedback. The lack of a query language model in the existing work on the language modeling approach has created difficulties in both understanding and extending the approach. In particular, it makes it hard or unnatural to incorporate feedback,

[1] Even the KL-divergence function has another form, since it is not symmetric.

a very important retrieval technique. Our introduction of a query language model is an important step toward more powerful retrieval models based on language modeling.

The simplest generative model for a query is one that generates a query directly from the query topic language model. Let $\mathbf{q} = q_1 q_2 ... q_n$ be a query and let $\theta_{\mathbf{q}}$ be the query topic language model. We have

$$p(\mathbf{q}) = \prod_i p(q_i \mid \theta_{\mathbf{q}})$$

To estimate $\theta_{\mathbf{q}}$ based on an observed query q, we can use the following maximum likelihood estimator

$$\widehat{\theta}_{\mathbf{q}} = \arg\max_{\theta_{\mathbf{q}}} \prod_i p(q_i \mid \theta_{\mathbf{q}})$$

We have seen in Section 6.1 that when using this model, we essentially have the popular query likelihood model.

In the next sections, we will describe several different ways of estimating a query topic language model. We will show that a better query language model can be estimated by exploiting a mixture model to model queries of different amounts of noise. We will propose two different methods for estimating a query model based on feedback documents. Experiment results show that both are effective for pseudo feedback; the updated query model can perform significantly better than the original model.

6.4 Query Model Distillation

It has been observed that the length and type of query may affect the retrieval performance significantly. For example, long verbose queries tend to perform better than short keyword-like queries, and special heuristics have been proposed to improve retrieval performance on short queries (Kwok and Chan, 98).

The maximum likelihood estimate of the query language model discussed above may not be able to effectively handle queries with different amounts of noise, as we have discussed in Chapter 4. To see the problem, recall that the query likelihood ranking formula is essentially a sum of weight for each query term matched by a document. The weight is higher if a query term is more frequent in the query text, which means that if a document matches only a common term in the query, its score will be higher than if it matched a rare term in the query, but a very common word is unlikely to be important for a query. On the other hand, if a document does not match a common term, the score will be significantly

penalized. We know that a long verbose query will likely have many more common words than a short query, so the query likelihood model is likely to over-emphasize a common word in the case of a long verbose query.

In Chapter 5, we have addressed this problem with a two-stage language model, in which one is explicitly modeling the query noise. In the KL-divergence model, a natural way to address this problem is to allow a verbose query to be generated by a mixture model where the query topic model is mixed with some general English model such as a collection language model. That is, we will assume that a query is generated by picking either a topic word or a word from general English.

Formally, let $p(w \mid \theta_{\mathbf{q}})$ be the query topic model and p_c be a collection unigram model. The mixture model is given by $p(w) = (1-\lambda)p(w \mid \theta_{\mathbf{q}}) + \lambda p_c(w)$

The mixture weight parameter λ is related to the verbosity of the query. For verbose queries, we expect that λ will be larger than for short queries. Intuitively, λ can be interpreted as indicating the amount of "noise" when generating a query.

In order to exploit this simple mixture model in retrieval, we first estimate $\widehat{\theta}_{\mathbf{q}}$ for any given query, and then compute the KL-divergence between $p(w \mid \widehat{\theta}_{\mathbf{q}})$ and $p(w|\widehat{\theta}_{\mathbf{d}})$, where $\widehat{\theta}_{\mathbf{d}}$ is the estimated document language model.

The estimation of $p(w \mid \widehat{\theta}_{\mathbf{q}})$ can be done by applying the EM algorithm, which results in the following updating formulas:

$$t^{(n)}(w) = \frac{(1-\lambda)p_{\lambda}^{(n)}(w \mid \widehat{\theta}_{\mathbf{q}})}{(1-\lambda)p_{\lambda}^{(n)}(w \mid \widehat{\theta}_{\mathbf{q}}) + \lambda p_c(w)}$$

$$p_{\lambda}^{(n+1)}(w \mid \widehat{\theta}_{\mathbf{q}}) = \frac{c(w)t^{(n)}(w)}{\sum_i c(w_i)t^{(n)}(w_i)}$$

Since we only need to go through all the query terms at each iteration, the EM algorithm is quite efficient. We now present some experiment results. They show that the mixture model generally performs better than the simple model for long verbose queries.

Intuitively, assuming a generative mixture model for queries is expected to help deal with queries of different amounts of noise. Our hypothesis is that with such a mixture model we can estimate a "distilled" query topic model that has excluded any possible "background noise" (i.e., common words in the query) from the model and thus reflects the true topic better than the maximum likelihood estimate. Since the distillation affects long verbose queries more than short keyword ones, we may expect it to improve the performance on

long TREC queries more significantly. To test this hypothesis, we compare the performance of the "distilled" query topic model with the maximum likelihood model estimated using the original query on two testing collections. One collection is AP89 with topics 1-50 and the other is the same web collection as used in the evaluation of smoothing methods. (See Section 4). These long queries are constructed based on all the text available in the original TREC description of the topics, with about 50 terms each on average.

A natural candidate for the background model is the collection unigram model. However, if we believe that the background model is supposed to model the common words in a query, then a model estimated using past queries might be a better background model. We have tried both on AP89 and found that a mixture of the past query model and the collection model performs the best. It appears that the model built using only the past queries is quite biased, since we have extremely sparse data, so smoothing with a collection model is reasonable. Such smoothing may become less important as we have more sample query data available. In our experiments on AP89, the linear interpolation smoothing coefficient is 0.2 for the collection model. For the WEB experiments, we have not tried the past query model yet, but we expect it to help as well. We only tried long verbose queries, so all the results reported here are only applicable for such queries. We expect the improvement to be less significant for short queries.

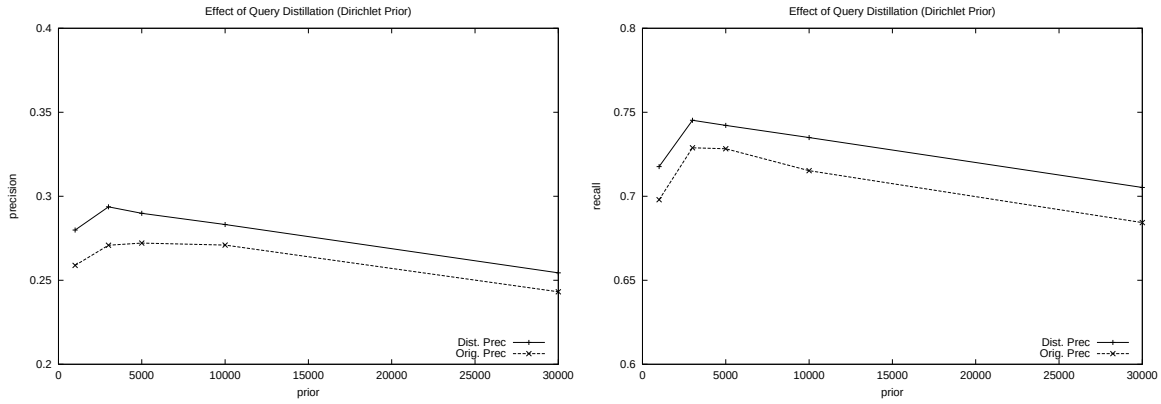

Figure 6.1: Average precision (left) and recall (right) for distilled query model and original query model on AP89.

Figure 6.1 shows that the distilled query indeed improves both recall and precision at all values of the smoothing prior parameter in the Dirichlet prior smoothing method for AP89. The distilled query model was estimated by setting the background noise coefficient λ to 0.5 (a relatively good setting for topics 1-50). Figure 6.2 shows a similar comparison for WEB. Again, we see that the distilled query model improves both recall and precision consistently.

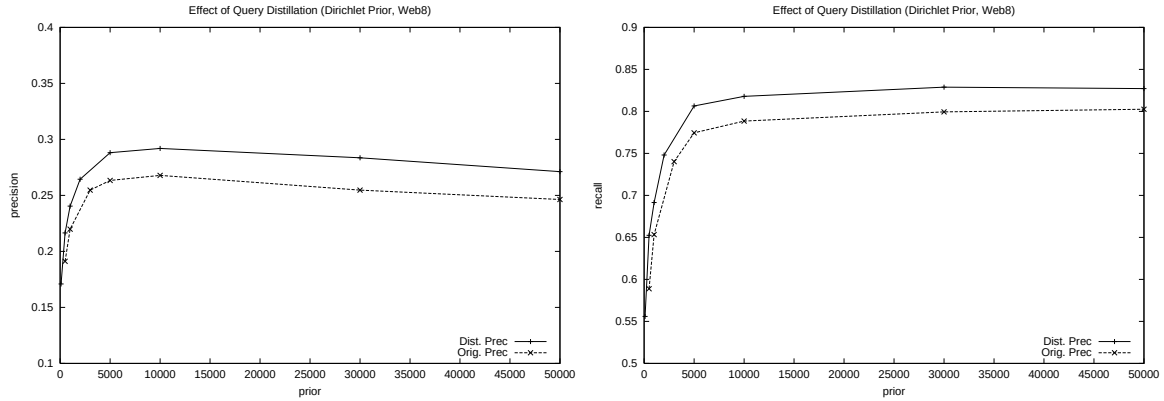

Figure 6.2: Average precision (left) and recall (right) for distilled query model and original query model on WEB.

However, this time, the optimal λ was 0.99 (for topics 401-450). More experiments are needed to understand how to set the noise coefficient λ. Presumably, the optimal value of λ is related to the query length and distribution properties of the query terms.

Figure 6.3 compares the average precision-recall curve of distilled query models with that of the original queries on both AP89 and WEB. We see that the distilled query is better than the original query at almost all recall points, though the effect is most clearly seen at high recall points.

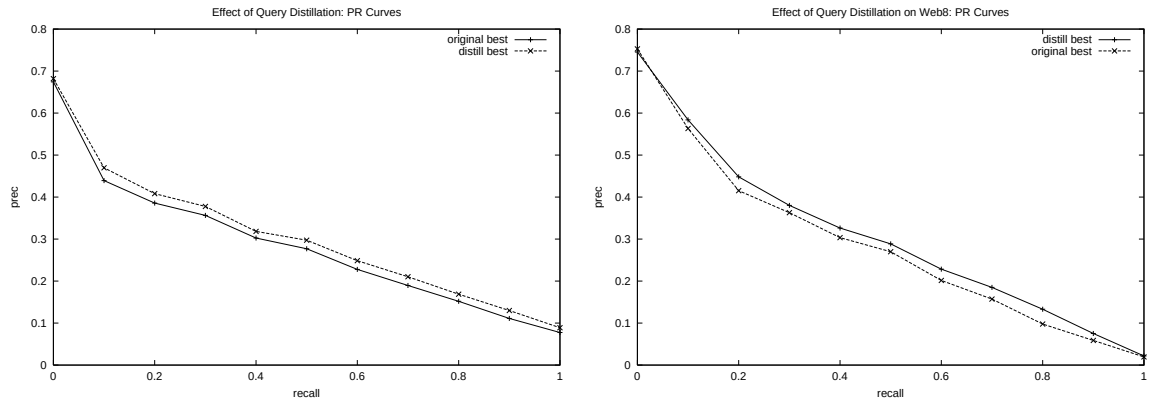

Figure 6.3: Precision-recall curves for distilled query model and original query model on AP89 (left) and WEB (right).

A similar distillation mixture model can also be applied to documents to extract a more focused document language model for each document. Since common words tend to have higher probabilities according to the background model, their probabilities in the focused model would be significantly discounted. As a result, the high probability words in the

focused model would be those content words that occur frequently in a document. Thus, such a distillation would achieve a similar effect as TF-IDF weighting of terms. Since the distillation does not depend on any query, it can be done at indexing time.

6.5 Model-based Feedback

Feedback can be regarded as a learning component in a TR system, in which the system learns from example relevant documents to improve the ranking function. [2]

There are two different scenarios for feedback: relevance feedback and pseudo feedback, but the underlying technique is generally similar. The difference between relevance feedback and pseudo feedback is where the example documents come from. In relevance feedback, they are the documents that a user actually judged as relevant, while in pseudo feedback they are usually some top ranked documents from a previously retrieved result (Evans and Lefferts, 1994; Buckley, 1995; Xu and Croft, 1996). Feedback, especially relevance feedback, often improves the performance quite significantly. Indeed, pseudo feedback is one of the most successful heuristics used in the traditional retrieval models to improve the performance of automatic ad hoc retrieval. The typical feedback procedure in a traditional model involves extracting terms from the example documents and adding the terms to the query, i.e., expanding the query with terms from the example documents. Not surprisingly, the same procedure has also been used for feedback in the language modeling approach.

For example, a ratio approach was proposed in (Ponte, 1998), which selects terms with a high distribution in the feedback documents, but low distribution according to the collection language model. The approach performs similarly to the Rocchio method (Rocchio, 1971) when very few relevant documents are used, but is significantly better than Rocchio when more relevant documents are used. The pseudo relevance feedback results are also very promising, and significantly better than the results of using the baseline language modeling approach (Ponte, 1998). However, the ratio approach is conceptually restricted to the view of query as a set of terms, so cannot be applied to the more general case when the query is considered as a *sequence* of terms and the frequency information of a query term is considered. Also, the number of terms needs to be determined heuristically. Miller and others treat feedback as essentially expanding the original query with all terms in the feedback documents (Miller et al., 1999). Terms are pooled into bins by the number of

[2] A more general discussion of learning in TR can be found in (Fuhr, 1993).

feedback documents in which they occur, and for each bin, a different transition probability in the HMM is heuristically estimated. [3] The performance of such a feedback technique is reported to be quite promising and robust. However, the interpretation of a query both as a text (generated by an HMM) and as a set of terms is conceptually inconsistent. It also involves heuristic adjustment of transition probabilities by incorporating document frequency to "filter" out the high frequency words. In (Ng, 2000), the query likelihood ranking criterion was motivated based on the ratio of document likelihood, though it is not very clear what the underlying generative model is. In terms of ranking, it is essentially similar to the simple HMM used in (Miller et al., 1999). There are, however, two interesting ideas about feedback in (Ng, 2000). First, a feedback criterion based on the optimization of the scores of feedback documents is used, though the criterion turns out to be actually very similar to the ratio approach used in (Ponte and Croft, 1998). Second, a threshold for the number of selected terms is heuristically derived from the score optimization criterion. This approach is also reported to be effective (Ng, 2000). But it shares the same problem of inconsistency in interpreting the query as in (Miller et al., 1999).

Thus, although the previous work on feedback has been successful empirically, it is all essentially based on query expansion. But such an *expansion-based* feedback strategy is generally not very compatible with the essence of the language modeling approach – model estimation. As a result, the expanded query usually has to be interpreted in a *different* way than the original query is. This is in contrast to the natural way of performing feedback in the classical relevance-based probabilistic model, such as the binary independence model (Robertson and Sparck Jones, 1976).

We propose a *model-based* approach to feedback that can be incorporated into the KL-divergence retrieval model. The model-based approach to feedback is actually not new; indeed, it is the essence of the classical probabilistic model (Robertson and Sparck Jones, 1976). However, it has been unclear how we can do it with the query likelihood ranking function used in most existing work on the language modeling approach. The KL-divergence retrieval model is a more general way to exploit language modeling for information retrieval. It covers the query likelihood approach as a special case, so it can be regarded as a natural extension. Since it involves the estimation of both a query language model and a document language model, language modeling techniques can be incorporated into a retrieval model more flexibly. In particular, the query language model can be compared to the relevance model in the classical probabilistic retrieval model, and thus

[3] As a result, the smoothing is no longer equivalent to the simple linear interpolation as it is in their basic HMM.

feedback can be performed as re-estimating the query model based on feedback documents. We propose two different criteria for re-estimating the query model:

1. *Generative model of feedback documents:* Assume a generative model for feedback documents and estimate the query topic model (parameters of the generative model) based on the observed feedback documents (using the maximum likelihood method or any other method). We consider a mixture model with the collection language model as one component and the query topic model the other.

2. *Divergence/risk minimization over feedback documents:* The estimate of the query model is one that has a minimum average KL-divergence to all feedback documents.

It is worth mentioning that the Markov chain query expansion method, when applied to a set of feedback documents, can be regarded as a model-based approach to feedback, though it is meant to be a translation model (Lafferty and Zhai, 2001a). The relevance model estimation method proposed in (Lavrenko and Croft, 2001) can also be used to estimate a richer query model based on feedback documents. Both approaches rely on the query words to focus the model, and this is different from the approaches proposed here. We estimate a feedback model solely based on the feedback documents, and then interpolate the model with an existing query model. Another related work is (Hiemstra, 2001), in which feedback documents are used to re-estimate the smoothing parameters in the query likelihood retrieval function. In effect, it is similar to query term reweighting in a traditional retrieval model, so it really has not fully taken advantage of the feedback documents (e.g., no new terms are introduced to enhance a query).

The KL-divergence model supports model-based feedback very naturally. Specifically, when feedback documents are available, we basically face the problem of re-estimating the query model based on the feedback information plus the original query text. A simple (but general) way to achieve this is to assume that the feedback documents are the extra data that help *smooth* the original query language model. In this thesis, we explore the simplest smoothing method based on linear interpolation. Specifically, let $\widehat{\theta}_{\mathbf{q}}$ be the estimated original query model and $\widehat{\theta}_{\mathcal{F}}$ be an estimated feedback query model based on feedback documents $\mathcal{F} = (d_1, d_2, ..., d_n)$, which can be the documents judged to be relevant by a user (as in the case of relevance feedback) or the top documents from an initial retrieval (as in the case of pseudo relevance feedback). Then, our new query model $\widehat{\theta}'_{\mathbf{q}}$ would be

$$\widehat{\theta}'_{\mathbf{q}} = (1-\alpha)\widehat{\theta}_{\mathbf{q}} + \alpha\widehat{\theta}_{\mathcal{F}}$$

where, α controls the influence of the feedback model. We now describe two very different strategies for estimating $\widehat{\theta}_{\mathcal{F}}$ based on feedback documents.

6.5.1 Generative Model of Feedback Documents

A natural way to estimate a feedback query model ($\widehat{\theta}_{\mathcal{F}}$) is to assume that the feedback documents are generated by a probabilistic model $p(\mathcal{F} \mid \theta)$, where θ is the parameter of our query model. Then the problem of estimating a query model becomes a standard problem of parameter estimation given the observed $\mathcal{F}$. For example, using the maximum likelihood estimator, we have

$$\widehat{\theta}_{\mathcal{F}} = \arg\max_{\theta} p(\mathcal{F} \mid \theta)$$

A Bayesian estimator can also be used if we can specify a meaningful prior, but we will only consider the maximum likelihood estimator in this paper. The simplest generative model is a unigram language model, which generates each word in $\mathcal{F}$ independently according to θ. That is,

$$p(\mathcal{F} \mid \theta) = \prod_i \prod_w p(w \mid \theta)^{c(w;d_i)}$$

where $c(w; d_i)$ is the count of word w in document d_i. This simple model would be reasonable if our feedback documents only contain relevant information. However, most documents probably also contain background information or even non-relevant topics. A more reasonable model would be a mixture model that "generates" a feedback document by mixing the query topic model with a collection language model. That is, a document is generated by picking a word using either the query topic model $p(w \mid \theta)$ or the collection language model $p(w \mid \mathcal{C})$. Without other knowledge, the collection language model is a reasonable approximation of the model for generating the irrelevant content in a feedback document.

Under this simple mixture model, the log-likelihood of feedback documents is

$$\log p(\mathcal{F} \mid \theta) = \sum_i \sum_w c(w; d_i) \log((1-\lambda)p(w \mid \theta) + \lambda p(w \mid \mathcal{C}))$$

Note that if both λ and θ are to be estimated, then the maximum likelihood estimate of λ will be zero and our mixture model will be degenerated to the simple unigram model. Intuitively, however, we would like to have a non-zero λ, and it can be interpreted as indicating the amount of background "noise" when generating a document. Thus, we will

set λ to some constant and estimate only θ, which can be done by using the EM algorithm (Dempster et al., 1977). The EM algorithm leads to the following updating formulas for $p_\lambda(w \mid \widehat{\theta}_{\mathcal{F}})$:

$$t^{(n)}(w) = \frac{(1-\lambda)p_\lambda^{(n)}(w \mid \theta_{\mathcal{F}})}{(1-\lambda)p_\lambda^{(n)}(w \mid \theta_{\mathcal{F}}) + \lambda p(w \mid \mathcal{C})}$$

$$p_\lambda^{(n+1)}(w \mid \theta_{\mathcal{F}}) = \frac{\sum_{j=1}^{n} c(w; \mathbf{d}_j) t^{(n)}(w)}{\sum_i \sum_{j=1}^{n} c(w_i; \mathbf{d}_j) t^{(n)}(w_i)}$$

Intuitively, when estimating the query model, we are trying to "purify" the document by eliminating some "background noise." Thus, the estimated query model will generally be concentrated on words that are common in the feedback document set, but not very common according to the collection language model $p(. \mid \mathcal{C})$. This is exactly what most traditional feedback methods, such as the Rocchio method (Rocchio, 1971), try to capture. Each iteration of the EM algorithm involves iterating over all the distinct terms in all the feedback documents, thus the complexity is $O(|F|L)$, where $|F|$ is the number of feedback documents and L is the average number of distinct terms in a feedback document. This is about the same as other traditional algorithms such as Rocchio. Since we generally need at least several iterations, the actual overall complexity would still be higher than Rocchio, but in practice, the difference will probably not matter that much, since $|F|$ is typically small.

To score a document $\mathbf{d}$ using the estimated query model $\widehat{\theta}_{\mathcal{F}}$, we first interpolate it with the original query model $\widehat{\theta}_{\mathbf{q}}$ to obtain an updated query model $\widehat{\theta}_{\mathbf{q}'}$, and then compute the KL-divergence between $p(w \mid \widehat{\theta}_{\mathbf{q}'})$ and $p(w \mid \widehat{\theta}_{\mathbf{d}})$, where $\widehat{\theta}_{\mathbf{d}}$ is an estimated smoothed document language model.

6.5.2 Divergence Minimization over Feedback Documents

A different strategy for estimating a query model based on feedback documents is to minimize the empirical KL-divergence of the model over the feedback documents. This strategy for feedback is consistent with the risk minimization framework, from which the KL-divergence model is derived.

Let $\mathcal{F} = (\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_n)$ be a set of feedback documents. We define the empirical KL-divergence of a query model θ over $\mathcal{F}$ as

$$D_e(\theta;\mathcal{F}) \;=\; \frac{1}{|\mathcal{F}|}\sum_{i=1}^{n} D(\theta||\widehat{\theta}_{\mathbf{d}_i})$$

i.e., the average of the KL-divergence with respect to the empirical word distribution of each document ($\widehat{\theta}_{\mathbf{d}_i}$).

Intuitively, if we estimate the query model by minimizing this average divergence, we will have a query model that, if used to score documents, will give us the best average score (or equivalently, minimum average divergence) over the feedback documents. The estimated query model will be as close to each feedback document model as possible. However, since feedback documents share many common words (due to the language and domain characteristics), such a query model may be quite general, especially if we truncate the model for the sake of efficiency. One solution to this problem is to add a regulation condition to the divergence function. We can postulate that a query model is preferred if it incurs a greater divergence with respect to the collection model, which is an approximation of the language model for distracting or background contents.

Incorporating this condition, we end up with the following empirical divergence function of a feedback query model:

$$\begin{aligned} D_e(\theta;\mathcal{F},\mathcal{C}) &= \frac{1}{|\mathcal{F}|}\sum_{i=1}^{n} D(\theta||\widehat{\theta}_{\mathbf{d}_i}) - \lambda D(\theta||p(.\,|\,\mathcal{C})) \\ &= \sum_{w} p(w\,|\,\theta)\log p(w\,|\,\theta) - \\ &\quad (\frac{1}{|\mathcal{F}|}\sum_{i=1}^{n}\log p(w\,|\,\mathbf{d}_i) - \lambda\log p(w\,|\,\mathcal{C}))p(w\,|\,\theta) \end{aligned}$$

where $\lambda \in [0,1)$ is a weighting parameter, and $p(w\,|\,\mathcal{C})$ is the collection language model.

We easily recognize that the first term is essentially the entropy of θ (with a different sign) and the second term reflects our desire to have θ similar to our feedback document models, but dissimilar to the collection model. Thus minimizing this divergence is equivalent to maximizing the entropy of the model under our preference constraint encoded in the second term. This is very similar to the maximum entropy approach to parameter estimation.

Using the divergence minimization criterion, our estimate of the query model based on $\mathcal{F}$ is

$$\widehat{\theta}_{\mathcal{F}} = \arg\min_{\theta} D_e(\theta;\mathcal{F},\mathcal{C})$$

And $\widehat{\theta}_{\mathcal{F}}$ can be easily found analytically using the Lagrange multiplier approach. The solution is

$$p(w|\widehat{\theta}_{\mathcal{F}}) \propto e^{\frac{1}{1-\lambda}(\frac{1}{n}\sum_{i=1}^{n}\log p(w\,|\,\widehat{\theta}_{d_i})-\lambda\log p(w\,|\,\mathcal{C}))}$$

subject to a normalization constant. The computation of this solution is very efficient, as it only involves one iteration over the feedback documents; indeed, this is at the same level of complexity as Rocchio.

$\widehat{\theta}_{d_i}$ is a smoothed unigram language model estimated based on document d_i. We see that a query model estimated according to divergence minimization tends to assign a high probability to words that are common in the feedback documents, but not common according to the collection language model. The parameter λ controls our weight on the collection language model. Similar to the λ in the collection mixture model, when λ is set to zero, the effect of collection language model is completely ignored, thus we have a query model that strictly minimizes the divergence over the feedback documents.

Again, to exploit $\widehat{\theta}_{\mathcal{F}}$ in our KL-divergence retrieval model, we first interpolate it with the original query model $\widehat{\theta}_{\mathbf{q}}$ to obtain an updated model $\widehat{\theta}_{\mathbf{q}'}$, and then score a document $\mathbf{d}$ by $D(\widehat{\theta}_{\mathbf{q}'}||\widehat{\theta}_{\mathbf{d}})$.

We evaluate these two model-based feedback methods for performing pseudo feedback on three different testing collections. Preliminary evaluation results show that both methods are effective for pseudo feedback and the updated query model may perform significantly better than the simple non-feedback model.

6.5.3 Evaluation of Model-based Pseudo Feedback

In this section, we report the preliminary results of using the proposed two model-based feedback methods for pseudo feedback on the following three testing collections:

1. AP88+89 with topics 101-150. This is the same collection as used in (Lavrenko and Croft, 2001), and will be labeled as AP88-89.

2. Trec Disk4&5-CR with topics 401-450. This is the official TREC8 ad hoc task collection, and will be labeled as TREC8.

3. Trec8 small web collection with topics 401-450. This is the official TREC8 small web task collection, and will be labeled as WEB.

We fixed the document language model in order to focus on exploring different ways of estimating query models based on feedback documents. Specifically, we used the Dirichlet prior method (with a prior of 1,000) for estimating a document language model in all the experiments. An appropriate way of evaluating a feedback method would be to consider both relevance feedback and pseudo (or blind) feedback, but as a first step, we only consider pseudo feedback. In all experiments, we take the top 10 documents from a set of previously retrieved results obtained using the basic query likelihood ranking function and the Dirichlet smoothing method with a prior of $1,000$. We compare the query models estimated using the collection mixture model method and the divergence minimization method, and vary both the interpolation parameter (α) and the feedback model estimation parameters (the $\lambda's$).

In all cases, we use only the titles of the topic description, since they are closer to the actual queries used in real applications, and also, feedback is expected to be most useful for short queries. We have done minimum preprocessing of documents and queries; the only tokenization performed is stemming (using a porter-stemmer). No stopword list is applied. We believe that with appropriate probabilistic modeling, stop words can be effectively down-weighted.

Effect of feedback

In order to see the effect of feedback, we compare the feedback results with the baseline non-feedback results. In general, we find that, with appropriate parameter settings, both feedback techniques that we proposed can be very effective. For example, the best feedback results from each method (with λ fixed to 0.5) are compared with the baseline performance in Figure 6.4 and Table 6.1. We notice that the average precision and recall are consistently improved by performing feedback. The increase in average precision is larger than 10% in most cases. We can also notice that the initial precision of feedback results is slightly decreased in almost all cases. Given that all the top ten documents may not be relevant, this is not very surprising, as the initial precision is very sensitive to the ranking of one particular document on the top, while our goal is to improve the overall ranking of documents. Note that despite the decrease in initial precision, the precision at 10 documents and 20 documents is consistently improved. It is interesting that the improvement on AP88-89 is much greater than that on TREC8 and WEB. This seems to be true for both approaches and also true for the Rocchio approach to be discussed below, thus it may suggest that feedback on AP88-89 is somehow "easier" than that on TREC8 or WEB (perhaps, because

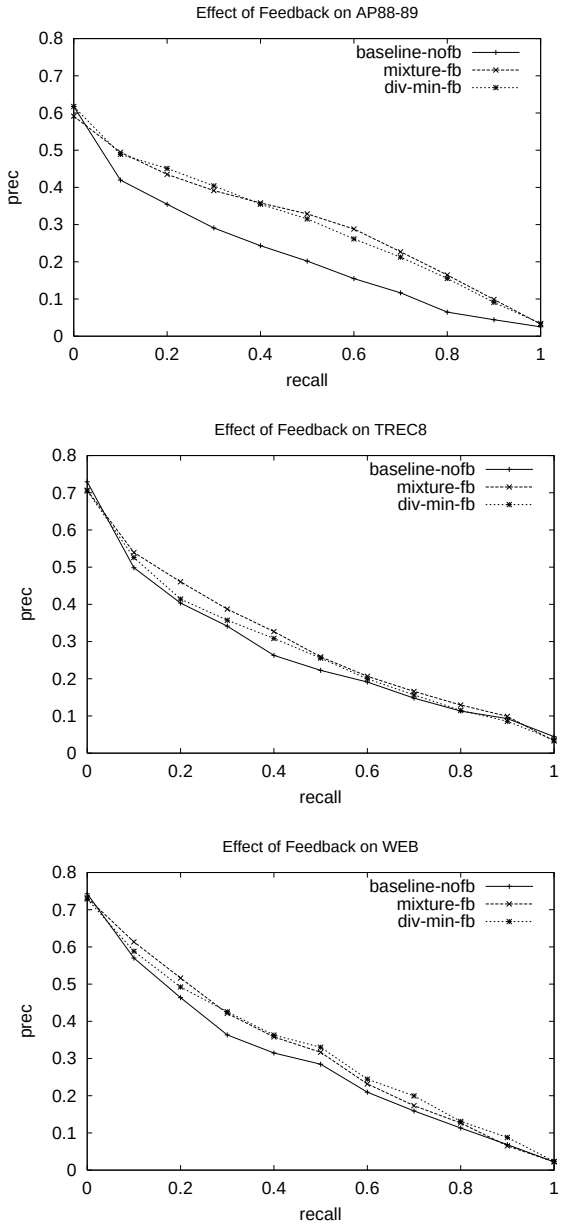

Figure 6.4: Effect of feedback on AP88-89 (top), TREC8 (middle), and WEB (bottom). In each picture, the two feedback methods are compared with the baseline simple language modeling approach (no feedback).

of the homogeneity of documents and/or a higher density of relevant documents). Further experiments and analysis are needed to understand this better.

In Table 6.2, we compare our feedback results with that of a tuned Rocchio approach with TF-IDF weighting, both using the top 10 documents for feedback. The TF formula

Collection		Simple LM	Mixture FB ($\lambda = 0.5$)	Improv.	DM FB ($\lambda = 0.5$)	Improv.
AP88-89	AvgPr	0.210	0.296 ($\alpha = 0.9$)	$+41\%$	0.295 ($\alpha = 1.0$)	$+40\%$
	InitPr	0.617	0.591 ($\alpha = 0.9$)	-4%	0.617 ($\alpha = 1.0$)	$+0\%$
	Prec@10	0.372	0.398 ($\alpha = 0.9$)	$+7\%$	0.406 ($\alpha = 1.0$)	$+2\%$
	Prec@20	0.340	0.389 ($\alpha = 0.9$)	$+14\%$	0.408 ($\alpha = 1.0$)	$+20\%$
	Recall	3067/4805	3888/4805 ($\alpha = 0.9$)	$+27\%$	3665/4805 ($\alpha = 1.0$)	$+19\%$
TREC8	AvgPr	0.256	0.282 ($\alpha = 0.7$)	$+10\%$	0.269 ($\alpha = 0.9$)	$+5\%$
	InitPr	0.729	0.707 ($\alpha = 0.7$)	-3%	0.705 ($\alpha = 0.9$)	-3%
	Prec@10	0.448	0.452 ($\alpha = 0.7$)	$+1\%$	0.446 ($\alpha = 0.9$)	-0%
	Prec@20	0.397	0.420 ($\alpha = 0.7$)	$+6\%$	0.410 ($\alpha = 0.9$)	$+3\%$
	Recall	2853/4728	3160/4728 ($\alpha = 0.7$)	$+11\%$	3129/4728 ($\alpha = 0.9$)	$+10\%$
WEB	AvgPr	0.281	0.306 ($\alpha = 0.5$)	$+9\%$	0.312 ($\alpha = 0.7$)	$+11\%$
	InitPr	0.742	0.732($\alpha = 0.5$)	-1%	0.728($\alpha = 0.7$)	-2%
	Prec@10	0.426	0.462 ($\alpha = 0.5$)	$+8\%$	0.464 ($\alpha = 0.7$)	$+9\%$
	Prec@20	0.358	0.387 ($\alpha = 0.5$)	$+8\%$	0.390 ($\alpha = 0.7$)	$+9\%$
	Recall	1755/2279	1758/2279 ($\alpha = 0.5$)	$+0\%$	1798/2279 ($\alpha = 0.7$)	$+2\%$

Table 6.1: Comparison of the basic language modeling method with model-based feedback methods. Columns three and five give the performance using the mixture model and divergence minimization respectively.

used is the one based on the BM25 retrieval formula with the same parameter settings as presented in (Robertson and Walker, 1999). We tuned the Rocchio method by varying the two main parameters in Rocchio (i.e., the coefficient and the number of terms); the reported results are the best results we obtained. Note that these Rocchio baseline results are actually very strong when compared with the published official TREC8 and WEB results (Voorhees and Harman, 2001), especially when considering that we used only title queries. We see that the two model-based feedback methods both perform better than Rocchio in terms of precision, though their recall is often slightly worse than Rocchio. The clear improvement in precision at 10 and 20 documents suggests that the language model feedback methods tend to do a better job at improving the ranking of top documents, while Rocchio seems to be better at low-level recalls. Another possibility is that we tuned the number of terms to use in the Rocchio method, but have not tuned the probability cutoff used in our methods, which affects the amount of terms to introduce for feedback. Indeed, in all the experiments, we truncated the estimated query model by ignoring all terms with a probability less than 0.001. It is reasonable to expect the recall to be improved when using a lower probability cutoff. Note that the precision can be expected to stay the same or increase as well when more terms are selected, because the extra terms generally have a very small probability,

Collection		Rocchio FB	Mixture FB ($\lambda = 0.5$)	Improv.	DM FB ($\lambda = 0.5$)	Improv.
AP88-89	AvgPr	0.291	0.296 ($\alpha = 0.9$)	$+2\%$	0.295 ($\alpha = 1.0$)	$+1\%$
	InitPr	0.566	0.591 ($\alpha = 0.9$)	$+4\%$	0.617 ($\alpha = 1.0$)	$+9\%$
	Prec@10	0.362	0.398 ($\alpha = 0.9$)	$+10\%$	0.406 ($\alpha = 1.0$)	$+12\%$
	Prec@20	0.363	0.389($\alpha = 0.9$)	$+7\%$	0.408 ($\alpha = 1.0$)	$+12\%$
	Recall	3729/4805	3888/4805 ($\alpha = 0.9$)	$+4\%$	3665/4805 ($\alpha = 1.0$)	-3%
TREC8	AvgPr	0.260	0.282 ($\alpha = 0.7$)	$+8\%$	0.269 ($\alpha = 0.9$)	$+3\%$
	InitPr	0.657	0.707 ($\alpha = 0.7$)	$+8\%$	0.705 ($\alpha = 0.9$)	$+7\%$
	Prec@10	0.434	0.452 ($\alpha = 0.7$)	$+4\%$	0.446 ($\alpha = 0.9$)	$+3\%$
	Prec@20	0.403	0.420 ($\alpha = 0.7$)	$+4\%$	0.410 ($\alpha = 0.9$)	$+2\%$
	Recall	3204/4728	3160/4728 ($\alpha = 0.7$)	-1%	3129/4728 ($\alpha = 0.9$)	-2%
WEB	AvgPr	0.271	0.306 ($\alpha = 0.5$)	$+13\%$	0.312 ($\alpha = 0.7$)	$+15\%$
	InitPr	0.600	0.732($\alpha = 0.5$)	$+22\%$	0.728 ($\alpha = 0.7$)	$+21\%$
	Prec@10	0.394	0.462($\alpha = 0.5$)	$+17\%$	0.464 ($\alpha = 0.7$)	$+15\%$
	Prec@20	0.341	0.387($\alpha = 0.5$)	$+13\%$	0.390 ($\alpha = 0.7$)	$+14\%$
	Recall	1826/2279	1758/2279 ($\alpha = 0.5$)	-4%	1798/2279 ($\alpha = 0.7$)	-2%

Table 6.2: Comparison of the Rocchio feedback method with model-based feedback methods. Columns three and five give the performance of using the mixture model and divergence minimization respectively.

so they will unlikely have a great impact on the ranking of documents with high scores. It would be interesting to test this hypothesis with experiments.

The comparison here is all based on some of the best feedback results. It is important that we also study how the feedback performance may be affected by the parameters in our model. We will first look at the interpolation coefficient α.

Influence of the interpolation coefficient α

Recall that we interpolate the estimated feedback query model with the original maximum likelihood model estimated based on the query text. The interpolation is controlled by a coefficient α. When $\alpha = 0$, we essentially end up using only the original model (i.e., no feedback), while if $\alpha = 1$, we completely ignore the original model and use only the estimated feedback model. In the actual experiments, we truncated the estimated feedback model by ignoring all terms with a probability lower than 0.001 and renormalized it before the interpolation.

Figure 6.5 shows how the feedback average precision varies according to the value of α. Each line represents a specific feedback model (estimated using either the mixture model

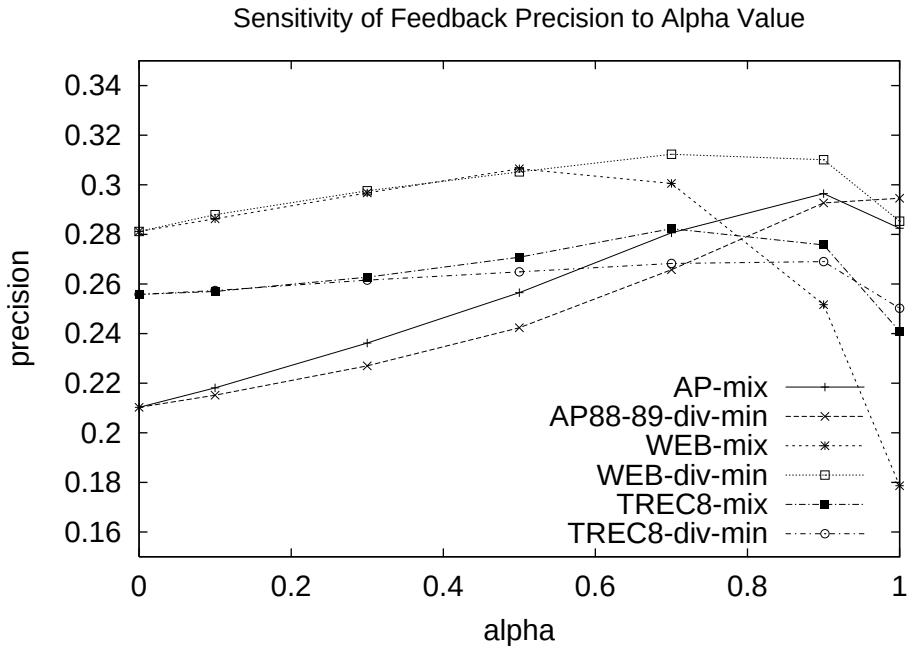

Figure 6.5: Influence of α value on precision. Lines represent different feedback models on different testing collections.

or the divergence minimization method) on a particular testing collection. Note that the precision at $\alpha = 0$ is actually the baseline non-feedback performance and the precision at $\alpha = 1$ is the performance given by using only the feedback model.

We see that the setting of α can affect the performance significantly. For example, on AP88-89, the feedback model alone is much better than the original query model, thus the optimal setting of α tends to be close to 1. On the other hand, on both TREC8 and WEB, the feedback model alone is much worse than that of the original query model, but when it is interpolated with the original query model appropriately, it can be much more effective than either model alone. This means that the two models complement each other very well. The original query model helps to focus on the topic, while the feedback model supplements it by suggesting related words. This is consistent with what people have observed with a traditional method like Rocchio, where the setting of the coefficient parameters also affects the performance significantly (Salton and Buckley, 1990). The optimal setting of α is somewhere between 0 and 1. It appears that it is usually safe to set α to a value close to, but smaller than, 0.5.

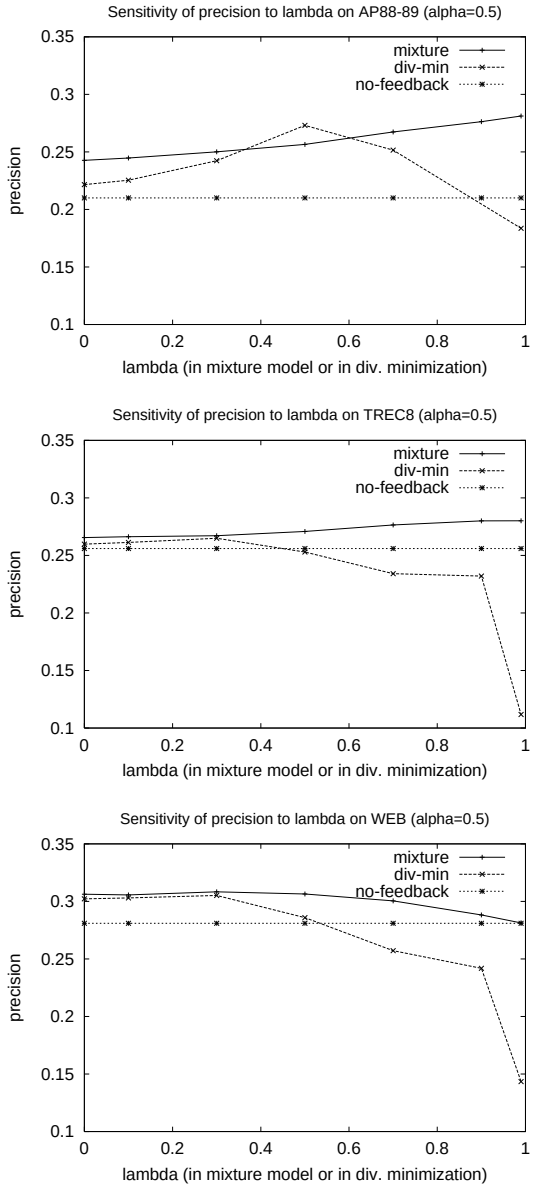

Figure 6.6: Sensitivity of precision to feedback model parameters on AP88-89 (top), TREC8 (middle), and WEB (bottom). In each picture, the horizontal line is the non-feedback performance, and the other two lines correspond to the two feedback methods respectively. Note that the x-axis means different λ for different methods.

Sensitivity of performance to feedback model parameter

We now look at the parameter in each feedback method. In the mixture model method, the parameter is the λ that controls the amount of "background noise" in the feedback docu-

ments, while in the divergence minimization method, the parameter is the λ that controls the influence of the collection language model. In both cases, λ indicates the extent to which the estimated query model should deviate from the collection language model. Although the two λ play a similar role conceptually, they affect the feedback performance in very different ways.

This difference can be seen from Figure 6.6, in which we show how the average precision changes according to different values of λ, when a fixed α (0.5) is used. Specifically, we can see that the performance is relatively insensitive to the setting of λ in the mixture model method, but can be quite sensitive to the setting of λ in the divergence minimization method. Indeed, with $\alpha = 0.5$, the mixture model performance is generally above the baseline, no matter what value we set λ to. However, the divergence minimization performance is only above the baseline when λ is small. When λ is big, the performance is extremely bad, significantly worse than the baseline performance. How to set these parameters automatically is an important topic for further research.

6.6 Conclusions and Further Work

In this chapter, we have explored a family of KL-divergence based retrieval models as special cases of the risk minimization framework. The KL-divergence models extend the query likelihood retrieval method through the incorporation of a query language model, which makes it possible to treat feedback naturally as query model updating.

We proposed two *model-based* methods for performing feedback in the language modeling approach to information retrieval. This is in contrast to the popular *expansion-based* feedback method used in most existing work. One advantage of the model-based approach is that it does not cause a conceptual inconsistency when interpreting the query in the retrieval model, and it explicitly treats the feedback process as a learning process.

In both methods proposed, the feedback documents are used to estimate a query model, which is then used to update the original query model with linear interpolation. The two methods differ in how they estimate the query model based on the feedback documents. The first method assumes the feedback documents are generated by a mixture model in which one component is the query topic model and the other is the collection language model. Given the observed feedback documents, the maximum likelihood method can be used to estimate a query topic model. The second method uses a completely different estimation criterion. It assumes that the estimated query model is the one that has the least average

KL-divergence from the empirical word distribution of each of the feedback documents. Both methods are relatively efficient to compute, though the second is more efficient than the first one; indeed, the second method is as efficient as the traditional Rocchio method.

The two methods were evaluated with three representative large retrieval collections. Results show that both methods are effective for feedback and perform better than the Rocchio method in terms of non-interpolated average precision. These results also show that better retrieval performance can be achieved through the use of more reasonable language models and better estimation methods. In particular, we have shown that the feedback documents can be exploited to improve our estimation of the query language model.

Analysis of results indicates that the performance can be sensitive to the settings of the interpolation coefficient α as well as the λ in each feedback method. It is relatively insensitive to the λ in the mixture model method, but very sensitive to the λ in the divergence minimization method. It appears that setting α to a value close to, but smaller than, 0.5 is good in most cases. A smaller λ (e.g., $\lambda = 0.3$) is probably appropriate for divergence minimization; while the λ in the mixture model method can be set to 0.5. Overall, the mixture model approach appears to be more robust.

Although these patterns are observed on feedback with only 10 documents, in some other experiments that have not been reported here, we found that with more feedback documents (e.g., 50), the sensitivity pattern appears to be basically similar to what we reported here, and the performance gain from feedback is usually even greater. Obviously, as we use more and more documents, the performance will eventually decrease. Indeed, that we have very little control over the true relevant examples is a serious drawback in experimenting with pseudo feedback only; it is often hard to tell if a bad feedback performance is due to a bad feedback technique or just bad luck with the feedback examples. An extreme case would be that the top 10 documents are all non-relevant, e.g., because of a bad initial ranking. Obviously, we cannot expect any feedback technique to gain much in this case. Thus, a very important direction for future work is to further test the proposed feedback techniques for relevance feedback, in which we will be able to examine the effectiveness of learning more closely.

Another interesting future research direction is to consider our confidence in assuming all the top 10 documents to be relevant. Intuitively, we would like to associate a relevance probability with each feedback document, so that the estimated query model will be affected more by those documents with a relatively high relevance probability. This relevance probability may come from the initial retrieval scores or may be estimated somehow

with the help of the original query text.

Finally, it would be very interesting to *derive* feedback language models directly using the risk minimization framework. One possibility is to assume that the relevance variable R (introduced in the relevance-based loss functions) is observed for the feedback documents, and to augment the observed variables to include these observed relevance values.

Chapter 7

Beyond Independent Topical Relevance

In this chapter, we present some preliminary research results in modeling the aspect retrieval problem within the risk minimization framework, which demonstrates how the risk minimization framework can go beyond the traditional notion of independent topical relevance. We study two different types of dependent loss functions. One is based on reducing the redundancy, or equivalently, emphasizing the novelty of new documents, which is an indirect way of increasing aspect coverage. The other is based on a direct modeling of aspects with language models. We present experimental results for both types of loss functions.

7.1 Introduction

While topical relevance is one of the most important factors in text retrieval, in real applications, a user often cares about other factors as well. For example, users generally would like to avoid seeing the same documents again or seeing similar when viewing the retrieval results. So, removing redundancy in documents would be desirable. Another example is when two or more documents together can satisfy a user's information need, but when judged individually, none of them is sufficiently relevant. An interesting discussion of the need for non-traditional ranking from the perspective of "optimal search behavior"can be found in (Varian, 1999).

To address factors like redundancy, it is necessary to break the *independent relevance* assumption (see Chapter 1). Most traditional retrieval models are based on such an independent relevance assumption, so are inadequate to address factors such as redundancy. Indeed, in these models, relevance is treated as a relationship between one *single* document

and one query. The risk minimization framework allows us to "encode" user factors that we would like to consider with the loss functions defined over a *set* of documents, so is general enough to model factors that may depend on more than one document.

We explore two different types of loss functions in the risk minimization framework, resulting in two different types of dependent retrieval models. The first type is Maximal Marginal Relevance (MMR) models (Carbonell and Goldstein, 1998), in which we model both *topical relevance* and *redundancy* of documents. Essentially, we want to retrieve relevant documents, and at the same time, minimize the chance that a user will see redundant documents as the user goes through the ranked list of documents. The second type of model is the Maximal Diverse Relevance (MDR) model, in which we model both *topical relevance* and *diversity* of documents. Here we assume that a user's goal is to retrieve documents that cover as many different relevant sub-topics or aspects as possible. Note that these two types of models are actually related. As we reduce the redundancy among documents, we can expect the coverage of the same aspect to be minimized and thus the coverage of potentially different aspects may be indirectly maximized. Similarly, as we maximize the aspect coverage, we are implicitly penalizing over-coverage of any single aspect, which should also help reduce the redundancy among documents. However, one could easily imagine cases where the goal of the MMR models and the goal of the MDR models are not consistent. We will discuss this more later.

For both types of loss functions, we explore different language models for capturing relevance, novelty, redundancy, and aspects. In general, relevance is usually captured through the KL-divergence of a query model and a document model. Novelty and redundancy are captured with either the KL-divergences between documents or mixture models. Aspects are modeled by generative aspect models.

Evaluation of such non-traditional models poses a challenge. First, most existing retrieval testing collections have been designed to evaluate traditional retrieval tasks; the relevance judgments are made independently for each document. Second, both redundancy and diversity in documents are extremely vague properties that are hard to quantify, and they tend to interact with relevance in the sense that over-emphasizing diversity may cause a decrease in relevance. Indeed, the optimal trade-off is a user factor, so is highly subjective and hard to quantify. In our research, we exploit the TREC interactive track data to set up an evaluation context for both types of models. The interactive track run under TREC studies the aspect retrieval problem, i.e., how to retrieve many different relevant aspects for a relatively general topic. Although the interactive track focuses on studying the interactive

retrieval problem, the generated test collection, including the aspect judgments, can be used to test our automatic ranking methods.

In the rest of this chapter, we first introduce the aspect retrieval task and then present each type of models along with its experimental results using the TREC interactive aspect retrieval data.

7.2 The Aspect Retrieval Task

7.2.1 Description

A regular retrieval task is often framed as retrieving relevant documents based on the assumption that the document is the information unit under consideration. However, a topic usually has some sub-topic structure, and involves different *aspects*. For example, a user who is looking for documents about tropical storms may be interested in identifying many different types or instances of tropical storms. Indeed, in real applications, users are often looking for particular aspects or as many relevant aspects as possible. The TREC interactive track has been designed to study such an aspect retrieval task in an interactive retrieval system. The following is an example query from TREC7 (number 392i), and the human assessors have found 35 different applications of robotics mentioned in the relevant documents to this query.

```
Number:   392i
Title:    robotics
Description:
        What are the applications of robotics in the world today?

Instances:
        In the time alloted, please find as many DIFFERENT applications of
        the sort described above as you can. Please save at least one
        document for EACH such DIFFERENT application.
        If one document discusses several such applications, then you
        need not save other documents that repeat those, since your goal
        is to identify as many DIFFERENT applications of the sort described
        above as possible.

Sample (relevant) aspects:

  1  'clean room' applications in healthcare & precision engineering
  2  spot-welding robotics
  3  controlling inventory - storage devices
  4  pipe-laying robots
  5  talking robot
```

```
  6  robots for loading & unloading memory tapes
  7  make & test materials with 15k different chemical compositions
  8  robot [telephone] operators
  9  robot cranes
... ...
```

Our goal is to address this problem with automatic approaches. Clearly, this would require a non-traditional ranking of documents, since ranking solely based on relevance would not be optimal. Both types of models that we propose can be expected to perform better than the standard relevance-based ranking. The MMR models may *indirectly* increase the aspect coverage as the redundancy is minimized, and the MDR models are directly aiming at improving the aspect coverage. Since the relevance/aspect judgments are available from the TREC interactive tracks, the aspect retrieval task provides an interesting and realistic setting in which to evaluate non-traditional ranking of documents.

7.2.2 Data Set

For three years (TREC-6, TREC-7, and TREC-8), the interactive track has used the same document collection – The Financial Times of London 1991-1994 collection (part of the TREC-7 adhoc collection). This collection is about 500MB and has 210,158 documents. The average document length is about 400 words (Voorhees and Harman, 2001). In each year, the interactive track task introduces six to eight new topics. We collected all 20 topics used in these three years. These topics are all formed by modifying the original ad hoc TREC topics. The modification is generally minimum – often just involving removing the "Narrative" section and adding an "Instance" section to explain what an *aspect* means for the topic.

The TREC (NIST) assessors would read a pool of documents submitted by TREC participants and identify a list of instances (i.e., aspects) and determine which documents contain or cover which instances. For example, for the sample topic 392i shown above, they identified 35 different aspects. The judgment for each document can be represented as a bit-vector with 35 bits, each indicating whether the document covers the corresponding aspect. Clearly, the length of such vectors varies according to different topics, ranging from 7 to 56 aspects, with an average of 20 aspects, in our data set. The number of judged relevant documents available also differs for different topics, with a range of 5 to 100 and an average of about 40 documents per topic. There are also some judgments of non-relevant documents. [1] We did not use that information, as we assume that any unjudged document is

[1]The total number of judgments, including both relevant and non-relevant documents, has a range of 22

non-relevant (i.e., covering no relevant aspect). This is a very strong assumption, which is unlikely to be true, since only the submitted (i.e., human-saved) results are judged, and the time that a user could spend working on those topics was limited. Thus, some documents down on the list are almost certainly relevant, but would be treated as non-relevant in our evaluation. We do not know how many documents are like this; our hope is that this biased evaluation will still be informative when *comparing different rankings*.

7.2.3 Evaluation

While the traditional relevance-based precision and recall measures can still be applied to the aspect retrieval task, they would not be sufficient, since what the user is actually interested in is not seeing as many relevant documents as possible, but seeing as many relevant *aspects* or *instances* as possible. Thus, we would need additional and more informative measures to evaluate the effectiveness of covering different aspects and avoiding redundant aspects.

Intuitively, three user factors are involved in the aspect retrieval task. The first is *topical relevance*. A good document should be topically relevant to the query topic. The second is *coverage of new aspects*. A good document should cover many different new aspects, i.e., many aspects that the documents ranked above it have not yet covered. And the third is *redundancy*. A good document should not be redundant with respect to the documents that are ranked above it, which presumably the user would have already seen before getting to this document. Ideally, a measure should cover all three factors. However, it is unclear how we can do that, as the optimal trade-off among the factors is not well-defined. Nevertheless, there are several interesting measures that we may consider using.

First, we can look at the *aspect coverage at different ranks*. That is, we count the number of different aspects at each of the ranks. Intuitively, this would tell us how well we cover the different aspects as the user goes down the ranked list of documents. A good ranking, by this measure, would allow a user to see all the aspects as early as possible. We can plot a curve to show how the aspect coverage changes as we consider different ranks. However, it is hard to come up with one single summary measure. A more serious problem with this measure is that it would not make much sense to average over different topics, since they may have different number of relevant aspects.

A modified, but similar measure is to examine the *aspect coverage at different recall*

to 243 documents and an average of 106 documents.

levels. That is, we look at the coverage at each different recall point. We can also plot a coverage-recall curve, just like the regular precision-recall curve. Also, similar to the relevance-based average precision, an aspect-based average coverage can also be computed by averaging the coverage at each point where a new aspect is retrieved/covered. Intuitively, it measures how fast we bring the new aspects to the user. We call this measure *Aspect Coverage* (AC).

The aspect coverage measure is actually a combined measure of relevance and aspects, since a good ranking must rank relevant documents above non-relevant documents, while at the same time ranking relevant documents that cover more aspects as early as possible. The underlying optimization problem is equivalent to the set-cover problem, which is known to be NP-hard. This means, even if we know the true number of aspects and know which document covers which aspects, it is still computationally complex to find an optimal ranking of documents under these measures.

One deficiency of the aspect coverage measure is that it does not consider redundancy – it does not penalize repeated covering of the same aspects. Presumably, we want to avoid covering the same aspects again while trying to cover as many new aspects as possible. Thus, we would also like to measure the redundancy, in addition to the aspect coverage. Unfortunately, the measurement of redundancy is tricky, because it is unclear whether a redundant relevant aspect should be treated as being better or worse than a non-relevant aspect. Actually, we do not even have the information about non-relevant aspects; we only know which document is a non-relevant document. So, let us first assume that we only work on relevant documents. Then, one reasonable measure of redundancy is to look at the percentage of all the relevant aspects that are distinct, i.e., the ratio of the number of distinct relevant aspects covered in a set of documents to the total number of relevant aspects. Intuitively, if all the aspects are unique, then there is no redundancy, so the ratio would be 1; otherwise, the ratio would be less than 1. To make this measure work on both relevant and non-relevant documents, we must decide what to do with the non-relevant documents. One possibility is to count each non-relevant document as one redundant aspect. Basically, we can assume a user can quickly judge a document as being non-relevant, making about the same effort as identifying a redundant relevant aspect. Essentially, we are measuring the number of aspects that a user needs to read before he/she sees a certain number of distinct aspects. In other words, the measure captures the uniqueness of the aspects that a user needs to read, so we call this measure *Aspect Uniqueness* (AU). To average over topics, we again look at the aspect uniqueness at different levels of aspect recall, similar to what we do with the aspect coverage.

The underlying optimization problem of aspect uniqueness is equivalent to the volume-cover problem, which is also known to be NP-hard. Thus, again, even if we know the true status of aspect coverage, it is still computationally complex to find an optimal ranking of documents. The complexity in finding an optimal ranking according to either aspect coverage or aspect uniqueness suggests that some kind of approximation would be inevitable in practice.

7.3 Maximal Marginal Relevance (MMR) Models

The idea of Maximal Marginal Relevance (MMR) ranking was first proposed and formalized in (Carbonell and Goldstein, 1998). It is based on the assumption that we need to consider not only the relevance value, but also the novelty (or equivalently, redundancy) in the presented documents when ranking a set of documents. Informally, given a set of previously selected documents, the next best document is one that is both *relevant* to the query topic and *different* from the already selected documents.

In the risk minimization framework, we can encode such preferences with a conditional loss function $l(d_k|d_1,...,d_{k-1},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))$ that "balances" the *relevance* value and the *redundancy* value of a document. Let $l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\theta_1,...,\theta^k)$ be such a loss function. The conditional risk is

$$r(d_k|d_1,...,d_{k-1},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) = \int_\Theta l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\theta_1,...,\theta_k)p(\theta|\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}})d\theta$$

If we assume that the parameters θ are concentrated at the mode $\hat{\theta} \equiv (\hat{\theta_Q},\{\hat{\theta}_i\}_{i=1}^k)$, then the posterior distribution is close to a delta function. In this simplified case, ranking based on the conditional risk is approximately equivalent to ranking based on the value of the loss function at the mode, i.e.,

$$r(d_k|d_1,...,d_{k-1},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) \overset{\text{rank}}{\approx} l_{MMR}(d_k|d_1,...,d_{k-1},\hat{\theta}_Q,\hat{\theta}_1,...,\hat{\theta}_k)$$

Technically, there could be many different ways to specify such a loss function. Indeed, deriving a well-motivated one is still an open research question.

In (Carbonell and Goldstein, 1998), the following MMR measure is given for re-ranking a pre-retrieved set of documents R. Given that the documents in S are already selected, the

next document D^* is given by

$$D^* = \arg\max_{D_i \in R \setminus S} [\lambda Sim_1(D_i, Q) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j)]$$

where λ is a parameter controlling the balance of relevance and redundancy, Sim_1 is a regular retrieval similarity function, and Sim_2 is a document similarity function that is supposed to capture redundancy.

A key component in a MMR model is the measure of the novelty of a candidate document, i.e., how much information contained in a candidate document is *new*. The challenge, however, is to come up with an appropriate combination of such a novelty measure with the regular relevance measure. In the formula given above, the novelty is measured *indirectly* by redundancy, i.e., the maximum similarity between the document and any of the old documents, and the novelty measure is combined with the relevance measure through linear interpolation. In the following section, we study both issues in the language modeling framework. First, we present two ways to measure the novelty of a document based on the KL-divergence measure and a mixture model. Then we discuss three possible approaches to combining novelty and relevance, corresponding to three different loss functions. The first approach assumes that there exist two *independent measures*, one for relevance and one for redundancy, and that the loss function is a combination of the two measures. The second approach assumes a *dynamic query model* that may depend on the documents that we have already picked. As we pick more and more documents, the query model would be more focused on the under-covered aspects. The third approach intends to estimate a "marginal" document model that is focused on representing the novel information contained in a candidate document, and then measure the relevance of the marginal document model with respect to the query model. We evaluate all the models using the aspect retrieval data.

7.3.1 Novelty/Redundancy Measures

Given a set of *reference language models* $\{\theta_1, ..., \theta_{k-1}\}$ and a model θ_k for a candidate document D_k, our goal is to define a novelty score S_N, such that $S_N(\theta_k; \theta_1, ..., \theta_{k-1})$ will indicate how much novel information document D_k contains. Intuitively, the novelty is correlated with how much information in D_k is redundant with respect to the reference models, which is further correlated with the similarity between documents. So, a redundancy measure can be converted to a novelty measure, though it may not reflect the novelty precisely, as a document could be redundant and cover new information.

Single Reference Topic Model

Let us start with the simplest case, where we have just one single reference model. Let θ_O and θ_N be the old (reference) model and the new document model respectively. How can we define $S_N(\theta_N; \theta_O)$? Intuitively, S_N should be related to the distance between θ_N and θ_O; the novelty in θ_N would be high if θ_N is different from θ_O. On the surface, it appears that, as a distance function, S_N should be symmetric. However, what we are really interested in here is actually an *asymmetric* distance. Specifically, we are more interested in how much of the information represented by θ_N is similar to that represented by θ_O, than vice versa. In other words, we are interested in whether all the information in the new document is new, but not in whether all the information in the old document has been duplicated in the new document. Now, let us discuss two different possibilities for S_N.

First, since we are working with unigram language models, the most natural distance measure would be the KL-divergence. It is asymmetric, but as discussed above, this is not necessarily a problem. Indeed, we should use $D(\theta_N||\theta_O)$ rather than $D(\theta_O||\theta_N)$ in order to measure the amount of new information represented by θ_N. Intuitively, $S_N(\theta_N; \theta_O) = D(\theta_N||\theta_O)$ tells us how inefficient it would be (e.g., in compression or encoding), if the true distribution is θ_N, but we approximate it with θ_O.

The KL-divergence measure would allow us to compare the redundancy of different documents, but it does not tell us what percentage of the new document is redundant/novel. We now introduce another novelty measure based on a simple mixture model. The basic idea is to assume a two-component mixture generative model for the new document, in which one component is the old reference topic model and the other is a background language model. Given the observed new document, we estimate the mixing weight for the background model (or the reference topic model), which can then serve as a measure of novelty (or redundancy). Intuitively, this weight indicates to what extent the new document can be explained by the background model as opposed to the reference topic model. A similar idea, but with three-component mixture models, has also been explored in (Zhang et al., 2002) to measure redundancy in information filtering.

Formally, let θ_B be a background language model, and λ be the mixing weight for the background model. The log-likelihood of a new document $D_k = w_1...w_n$ is

$$l(\lambda) = \sum_{i=1}^{n} \log((1-\lambda)p(w_i \,|\, \theta_O) + \lambda p(w_i \,|\, \theta_B))$$

and the estimated novelty score λ^* is given by

$$\begin{aligned}
\lambda^* &= \arg\max_{\lambda} l(\lambda) \\
&= \arg\max_{\lambda} \sum_{i=1}^{n} \log((1-\lambda)p(w_i \mid \theta_O) + \lambda p(w_i \mid \theta_B)) \\
&= \arg\max_{\lambda} \sum_{w} c(w, D_k) \log((1-\lambda)p(w \mid \theta_O) + \lambda p(w \mid \theta_B)) \\
&= \arg\max_{\lambda} \sum_{w} p(w \mid \theta_N) \log((1-\lambda)p(w \mid \theta_O) + \lambda p(w \mid \theta_B)) \\
&= \arg\min_{\lambda} D(\theta_N || \theta_{O,B}(\lambda))
\end{aligned}$$

where θ_N is the empirical word distribution of document D_k, and $\theta_{O,B}(\lambda)$ is the following mixture model:

$$p(w \mid \theta_{O,B}(\lambda)) = (1-\lambda)p(w \mid \theta_O) + \lambda p(w \mid \theta_B)$$

From the last equation, we see that our estimation of λ is essentially based on minimizing the KL-divergence of the empirical word distribution θ_N and the mixture model parameterized with λ. This is a standard estimation problem for a simple mixture model, so the maximum likelihood estimate can be easily computed using the EM algorithm.

In summary, the two basic novelty measures are:

1. KL-divergence: $S_N(\theta_N; \theta_O) = D(\theta_N || \theta_O)$
2. Mixture coefficient: $S_N(\theta_N; \theta_O) = \arg\min_{\lambda} D(\theta_N || \theta_{O,B}(\lambda))$

Multiple Reference Topic Models

When there is more than one reference topic model, there are two possibilities. One possibility is to compute an average model of all the reference topic models, and then use any of these two basic measures directly. The other possibility is to compute a novelty score with respect to each individual reference topic model, and then compute a single combined or summarized novelty score. The first method is straightforward, so we will discuss the second in some detail.

There are three obvious possibilities for combining the individual novelty scores:

1. Minimum distance: $S_N(\theta_k; \theta_1, ..., \theta_{k-1}) = \min_{1 \leq i \leq k-1} S_N(\theta_k; \theta_i)$
2. Maximum distance: $S_N(\theta_k; \theta_1, ..., \theta_{k-1}) = \max_{1 \leq i \leq k-1} S_N(\theta_k; \theta_i)$
3. Average distance: $S_N(\theta_k; \theta_1, ..., \theta_{k-1}) = \frac{1}{k-1} \sum_{i=1}^{k-1} S_N(\theta_k; \theta_i)$

The minimum distance essentially relies on the *closest* reference document to measure the redundancy, which implies a logical "OR" semantics. That is, as long as the new document is redundant with respect to one old document, it is judged as being redundant. It has the property of ensuring that the distance would be zero (most redundant) if the new document is identical to an old one.

The maximum distance does not appear to make much sense, since a document would be judged as not being redundant as long as it is very different from one old document, even though it is similar to all the others.

The average distance implies a logical "AND" semantics. That is, a new document would be judged as being redundant only when it is similar to *all* old documents. However, an identical duplicate document is *not* regarded as a most redundant document according to this measure. Instead, a document covering the whole set of old documents well will be judged as being most redundant. This is thus similar to the first strategy of computing an average model.

Therefore, we have six different novelty measures as shown in Table 7.1. Both minimum distance and average distance make sense, but considering our desire to avoid duplicated aspects, it appears that the minimum distance would be more appropriate, since it would ensure an exact duplicated aspect to be eliminated.

Basic Novelty Measure	**Aggregation**		
	Model aggregation (Model average)	Novelty aggregation	
		novelty minimum	novelty average
KL	AvgKL	KLMin	KLAvg
Mixture	AvgMix	MixMin	MixAvg

Table 7.1: Novelty measures based on language models.

Comparison of Novelty Measures

We compare all six novelty measures using the aspect retrieval task. In order to focus on the effectiveness of novelty detection and factor out the relevance, we consider the special task of re-ranking relevant documents. A novelty measure is expected to help bring novel aspects earlier in the ranked list by avoiding duplication of old aspects.

We rank a set of *relevant* documents using the following ranking procedure:

1. The first document is picked based on the regular relevance measure. Thus it is the same across all the methods.
2. Given that we have already picked k documents, the next document to be picked is the one that has the highest novelty value as determined by each of the six methods.
3. The process is repeated until we finish ranking all the documents.

We measure the ranking using both the Aspect-Coverage (AC) measure and the Aspect Uniqueness (AU) measure. The results are shown in Figure 7.1 and Figure 7.2 for AC and AU respectively.

From these curves, we can make several observations.

Ranking Method	Aspect Coverage			Aspect Uniqueness		
	Average	Recall=0.1	Recall=0.5	Average	Recall=0.1	Recall=0.5
Relevance	**1.941**	**3.060**	**2.132**	0.651	0.943	0.662
AvgKL	1.789	2.904	1.919	0.604	0.961	0.568
AvgMix	1.745	2.653	1.849	**0.773**	**0.99**	**0.783**
KLMin	1.861	2.939	2.032	0.641	0.971	0.630
KLAvg	1.832	2.919	1.998	0.625	0.964	0.604
MixMin	1.835	2.813	1.986	0.717	0.973	0.700
MixAvg	1.804	2.788	1.953	0.701	0.973	0.693

Table 7.2: Comparison of novelty measures and relevance ranking.

1. We see that the two measures have shown different orderings. While for the AU measure, the mixture model novelty measure performs significantly better than the KL-divergence measure, it is actually slightly worse than the KL-divergence measure by the AC measure. Indeed, the AvgMix measure performs significantly better than the relevance ranking baseline by the AU measure, but significantly worse than

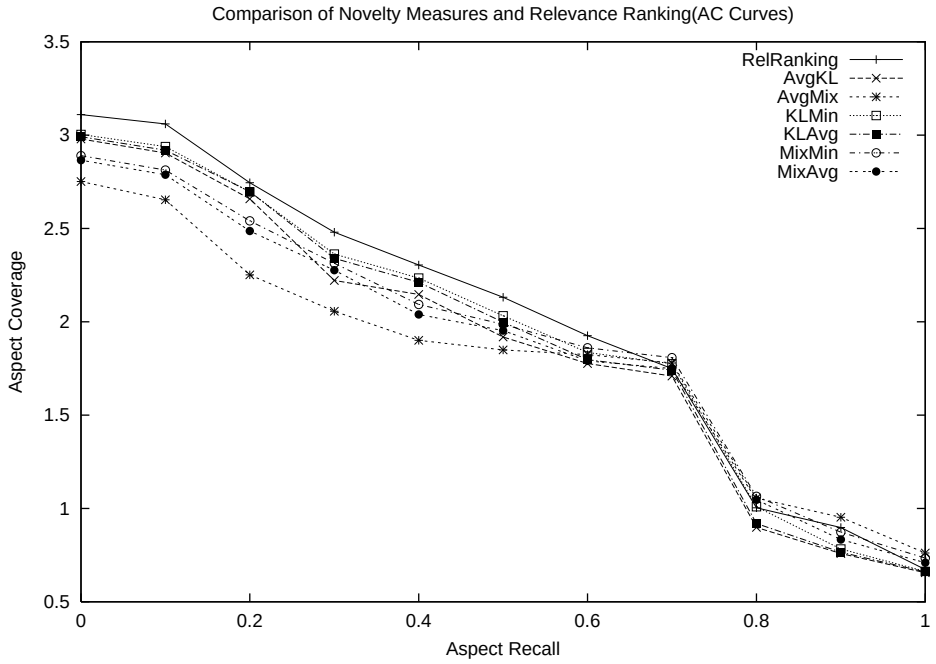

Figure 7.1: Comparison of the AC curves for the six novelty measures and the baseline relevance ranking (the solid line). All novelty measures are worse than the baseline.

the baseline by the AC measure. Since the AU measures the capability of removing redundancy, these observations suggest that removing redundancy does not necessarily result in improving the aspect coverage per document. It is possible that those documents that cover many new aspects also tend to overlap more with existing documents, so get down-weighted due to the high redundancy value.

2. The mixture model method is a better novelty (or equivalently redundancy) measure than the KL-divergence method. This can be seen clearly from Figure 7.2.

3. The two different methods to summarize the novelty as compared to a group of documents (i.e., the *minimum* method and the *average* method) do not appear to make much difference, but the minimum summary consistently performs slightly better than the average method. This can be seen more clearly from the results from Table 7.2, where we show the concrete figures at different points of the curves. Interestingly, by the AU measure, the *model average* approach works much better than the *novelty average* approach for the mixture model, though it is the opposite for the KL-divergence measure. This is somehow a surprise, since presumably, averaging the novelty score would achieve a similar effect to measuring the novelty based

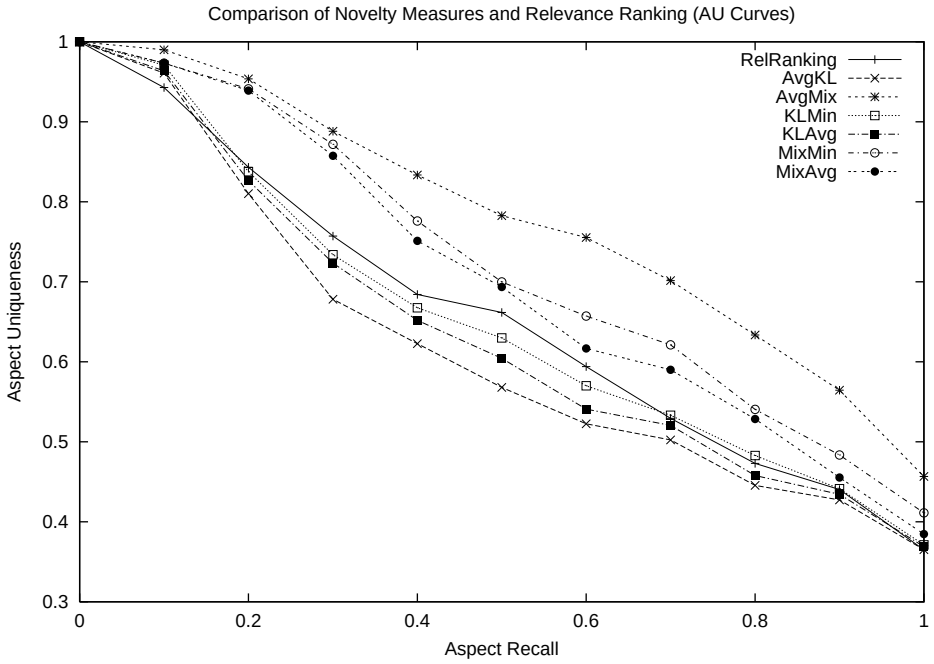

Figure 7.2: Comparison of the AU curves for the six novelty measures and the baseline relevance ranking (the solid line). All mixture model measures are better than the baseline, but all KL-divergence measures are worse.

on an average language model – both are trying to emphasize the "vote" from each document. One possible explanation is that when averaging the model, we obtain a model that is more smoothed and accurate, so the estimated mixture model coefficient can be expected to be accurate, whereas when each individual language model is used, the estimation of the coefficient may have higher variances, since we have more uncertainty on the individual models.[2]

4. From Table 7.2, note that the KL-divergence measure performs worse than the baseline ranking by all measures, except for the AU value at a low recall level (0.1), where it is always slightly better than the baseline. This initial improvement can be seen more clearly in Figure 7.2. This means, as a redundancy measure, it is only effective when measuring the redundancy of a document with respect to another or a few other documents. When the reference set is large, it appears to be ineffective.

5. Overall, AvgMix is the best by the AU measure, suggesting that it may be a reasonable redundancy measure. Unfortunately, by the AC measure, it is much worse

[2]Each individual model is smoothed using the Dirichlet prior method.

than all other measures. In general, we see that the two measures seem to compromise each other. One possible explanation is that when emphasizing on removing redundancy too much, we favor documents covering fewer aspects in general, which hurts the AC performance. It seems that even when we consider only relevant documents, the regular relevance measure is still helping us achieve a better aspect coverage. Thus, some combination of relevance and redundancy is necessary, even for re-ranking all relevant documents.

7.3.2 Combining Relevance and Novelty

An MMR loss function should be a combination of relevance measure and novelty measure to reflect our desire to retrieve a document that is both relevant and novel. Since the trade-off between relevance and novelty depends on the user model, in general, we should introduce a parameter to control this trade-off. We now discuss three possible ways of combining relevance and novelty, resulting in three types of MMR loss functions.

Direct Combination of relevance score and novelty score

Suppose we make the assumption that a relevance score and a novelty score can be computed *independently*. Then we can define our loss function as a direct combination of the two scores. Formally, let $S_R(\theta_k;\theta_Q)$ be any relevance scoring function and $S_N(\theta_k;\theta_1,...,\theta_{k-1})$ any novelty scoring function. An MMR loss function can be defined as a combination of the two scoring functions as follows.

$$l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\{\theta_i\}_1^{k-1}) = f(S_R(\theta_k;\theta_Q),S_N(\theta_k;\theta_1,...,\theta_{k-1}),\mu)$$

where $\mu \in [0,1]$ is a relevance-novelty trade-off parameter, such that

$$l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\theta_1,...,\theta_{k-1}) \overset{\text{rank}}{=} \begin{cases} S_R(\theta_k;\theta_Q) & \text{if } \mu = 0 \\ S_N(\theta_k;\theta_1,...,\theta_{k-1}) & \text{if } \mu = 1 \end{cases}$$

One possible such combination is a linear interpolation of S_R and S_N (Carbonell and Goldstein, 1998), given by

$$l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\{\theta_i\}_1^{k-1}) = (1-\mu)S_R(\theta_k;\theta_Q) + \mu S_N(\theta_k;\theta_1,...,\theta_{k-1})$$

In general, this loss function makes sense only when the range of the function S_R and that of S_N are comparable (e.g., when both S_R and S_N are KL-divergence). Thus, we can imagine a combination of a KL-divergence S_R with "AvgKL," "MinKL," or "KLAvg" for S_N. However, since the experiment results in the previous section have shown that the KL-divergence does not capture redundancy very well, such combinations cannot be expected to perform well.

Nevertheless, the KL-divergence is still a reasonable measure for relevance (S_R) as shown in many other relevance-based retrieval experiments. From the previous section, we also see that "AvgMin" is a reasonable measure of novelty (S_N). Thus, it would be desirable to combine them. Unfortunately, a direction interpolation of them would not make much sense, since they are not on the same scale. We note that the AvgMin estimate of S_N can be interpreted as the expected percent of "novel" information in the document with respect to the old documents, or the probability that a randomly picked word from the document represents *new* information. Thus, we may consider two probabilities associated with a document $\mathbf{d}$. One is the probability of relevance $p(Rel \mid \mathbf{d})$; the other is the probability that any word in the document carries some new information $p(New \mid \mathbf{d})$. This leads to the following general form of the loss function:

$$\begin{aligned} l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\{\theta_i\}_1^{k-1}) &= c_1 p(Rel \mid \mathbf{d}) p(New \mid \mathbf{d}) \\ &\quad + c_2 p(Rel \mid \mathbf{d})(1 - p(New \mid \mathbf{d})) \\ &\quad + c_3 (1 - p(Rel \mid \mathbf{d}) p(New \mid \mathbf{d}) \\ &\quad + c_4 (1 - p(Rel \mid \mathbf{d}))(1 - p(New \mid \mathbf{d})) \end{aligned}$$

where c_1, c_2, c_3, and c_4 are cost constants.

Since whether a non-relevant document carries any new information is not interesting to the user, we could reasonably assume that $c_3 = c_4$. Furthermore, we can also reasonably assume that there is no cost if the document is both relevant and (100%) new, i.e., $c_1 = 0$. Under these two assumptions, we have

$$l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\{\theta_i\}_1^{k-1}) = c_2 p(Rel \mid \mathbf{d})(1 - p(New \mid \mathbf{d})) + c_3 (1 - p(Rel \mid \mathbf{d})$$

For any reasonable loss function, both c_2 and c_3 should be some positive cost, and usually $c_3 > c_2$. In general, c_2 and c_3 may change according to k, or even $d_1, ..., d_{k-1}$, but in our studies, we assume that both are constants. Intuitively, c_2 is the cost of seeing a *relevant*, but *redundant* document, whereas c_3 is the cost of seeing a *non-relevant* document. Clearly,

when $c_2 = 0$, i.e., the user does not care about redundancy, and the loss function would be essentially based on the probability of relevance, just as what we would expect. Below we assume that $c_2 > 0$, which allows us to re-write the loss function in the following equivalent form for the purpose of ranking documents:

$$\begin{aligned} l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\{\theta_i\}_1^{k-1}) &= c_3 + c_2 p(Rel\,|\,\mathbf{d})(1 - \frac{c_3}{c_2} - p(New\,|\,\mathbf{d})) \\ &\stackrel{\text{rank}}{=} p(Rel\,|\,\mathbf{d})(1 - \frac{c_3}{c_2} - p(New\,|\,\mathbf{d})) \end{aligned}$$

We can see that, just as we hope, a higher $p(New\,|\,\mathbf{d})$ always helps reduce the loss, and when $\frac{c_3}{c_2} \geq 1$, a higher $p(Rel\,|\,\mathbf{d})$ also means a smaller loss. However, the amount of loss reduction is affected by the cost ratio $\frac{c_3}{c_2}$. This ratio indicates the relative importance of avoiding a non-relevant document vs. avoiding a relevant but redundant document. When the ratio is very large, i.e., $c_3 >> c_2$, the influence of $p(New\,|\,\mathbf{d})$ could be negligible. This means that when the user would not tolerate any non-relevant document, our optimal ranking would almost not be affected by the novelty of documents. In general, there would be a compromise between retrieving documents with new content and avoiding retrieving non-relevant documents.

One technical problem remains; we do not usually have $p(Rel\,|\,\mathbf{d})$ available when we score documents with the KL-divergence function. One possible solution is to consider ranking documents based on the query likelihood, i.e., $p(\mathbf{q}\,|\,\mathbf{d})$, which is equivalent to ranking based on the KL-divergence. Since $S_R = p(\mathbf{q}\,|\,\mathbf{d})$, we may further assume that $p(Rel\,|\,\mathbf{d})$ is *proportional* to $p(\mathbf{q}\,|\,\mathbf{d})$ by some constant. Under this assumption, the loss function can be rewritten as:

$$l_{MMR}(d_k|d_1,...,d_{k-1},\theta_Q,\{\theta_i\}_1^{k-1}) \stackrel{\text{rank}}{=} S_R(\theta_k;\theta_Q)(1 - \rho - S_N(\theta_k;\theta_1,...,\theta_{k-1}))$$

where $\rho = \frac{c_3}{c_2} \geq 1$, $S_R(\theta_k;\theta_Q) = p(\mathbf{q}\,|\,\mathbf{d})$ and $S_N(\theta_k;\theta_1,...,\theta_{k-1})$ is the estimated novelty coefficient using the mixture model method. We refer to this loss function as a *cost-based* combination of relevance and novelty.

A common deficiency in combining the relevance score and the novelty score is the assumption of *independent* measurement of relevance and novelty. In other words, we do not have a direct measure of relevance of the new information contained in a new document. Thus, a document formed by concatenating a seen (thus redundant) relevant document with a lot of new, but non-relevant, information may be ranked high, even though it is useless to the user. Next, we present two different types of loss functions that directly measure the relevance of the new information.

Dynamic query models

The basic idea of this type of loss function is to assume that the user's information need changes as the user browses through the documents according to the ranking order. In particular, the query model is assumed to depend on the documents that the user has already seen, and will focus more on the new aspects that have not been covered by the already picked/viewed documents. One way to implement this idea is to use the old reference language model as a query background model, i.e., the query is assumed to be generated by mixing the *known* information language model with a document language model. In this way, the documents would be discriminated primarily based on their capability of explaining the part of the query that cannot be accounted for by the "known model," rather than the whole query. Thus the documents will be discriminated mostly based on the relevance of the new information contained in each of them. This approach has already been mentioned at the end of Chapter 5.

Let θ_O be the old reference model which is the average of the models of all the previously picked documents ($\theta_1, ..., \theta_{k-1}$). Let $\theta_{O,k}$ be the following mixture model:

$$p(w \mid \theta_{O,k}) = (1 - \mu)p(w \mid \theta_k) + \mu p(w \mid \theta_O)$$

where μ indicates the weight on the reference model θ_O.

Then the loss function based on using the dynamic background model θ_O is given by

$$l_{MMR}(d_k|d_1, ..., d_{k-1}, \theta_Q, \theta_1, ..., \theta_k) = D(\theta_Q||\theta_{O,k})$$

While this method (called the *dynamic query background* method) can be expected to discriminate relevant documents based on the new information, it may not be very effective in ranking relevant documents above non-relevant documents. Indeed, as we allow the query background model to explain the query, a non-relevant document will be expected to get a reasonably high score. We now present a more reasonable loss function, where we explicitly estimate a new query model based on the current reference model θ_O.

We assume that the (original) query model is a mixture of θ_O and some new but unknown model θ_{Q_k}. This mixture model, denoted by θ_{O,Q_k}, is given by

$$p(w \mid \theta_{O,Q_k}) = (1 - \mu)p(w \mid \theta_{Q_k}) + \mu p(w \mid \theta_O)$$

where μ is a parameter that indicates how much known information we want to consider.

When $\mu = 0$, $\theta_{Q_k} = \theta_Q$. Now, given θ_Q, θ_O, and μ, we will estimate θ_{Q_k}, the dynamic query model for selecting the k-th document $\mathbf{d}_k$ by minimizing the KL-divergence:

$$\hat{\theta}_{Q_k} = \arg\min_{\theta_{Q_k}} D(\theta_Q || \theta_{O,Q_k})$$

The loss function based on this dynamic query model is

$$l_{MMR}(d_k | d_1, ..., d_{k-1}, \theta_Q, \theta_1, ..., \theta_k) = D(\hat{\theta}_{Q_k} || \theta_k)$$

We refer to this method as the *marginal query model* method.

Marginal document language models

The basic idea of this type of loss function is to estimate a language model that best represents the new information contained in a document. We imagine that a "residue" or "marginal" document may be generated by removing the redundancy in a given document. Thus, we would not use the novelty score directly; instead, we rely on some novelty measure to estimate a language model that corresponds to this imaginary marginal document.

To see how we could achieve such a goal in the language modeling approach, let us again consider the simplest case where we have only one single reference topic model θ_O. Suppose θ_B is a background language model and θ_N is the empirical word distribution of the new document. As discussed in Section 7.3.1, the estimated novelty is given by

$$S_N(\theta_N; \theta_O) = \arg\min_{\lambda} D(\theta_N || \theta_{O,B}(\lambda))$$

Note that $S_N(\theta_N; \theta_O) \in [0, 1]$ gives us an estimate of the fraction of novel information in the document. Now, let us assume that $S_N(\theta_N; \theta_O)$ is the true amount of novel information and the rest of the information can be explained by the old model θ_O very well. Our goal is to estimate a language model for the "identified amount of novel information." In order to estimate such a model, we consider a different mixture model that involves θ_O and an *unknown* marginal model θ_M (to be estimated). The mixing coefficient is fixed to $\lambda^* = S_N(\theta_N; \theta_O)$. Let $\theta_{O,M}(\lambda^*)$ be the following mixture model:

$$p(w \,|\, \theta_{O,M}(\lambda^*)) = (1 - \lambda^*) p(w \,|\, \theta_O) + \lambda^* p(w \,|\, \theta_M)$$

Then our estimate of θ_M is given by

$$\hat{\theta}_M = \arg\min_{\theta_M} D(\theta_N || \theta_{O,M}(\lambda^*))$$

Again the EM algorithm could be used here.

Once we have $\hat{\theta}_M$, we can simply rank documents based on their corresponding $\hat{\theta}_M$.

In summary, our idea is to first figure out how much new information the document d_k has, and then estimate a language model that represents the contained new information, and finally, score the document based on the KL-divergence of the marginal language model and the query model. In order to accommodate a flexible trade-off between relevance and novelty, we define our loss function as an interpolation of the regular relevance measure and this marginal relevance measure:

$$l_{MMR}(d_k|d_1, ..., d_{k-1}, \theta_Q, \theta_1, ..., \theta_k) = (1-\mu)D(\theta_Q||\theta_k) + \mu D(\theta_Q||\hat{\theta}_{M_k})$$

where $\hat{\theta}_{M_k}$ is an estimated marginal language model for document d_k. We see that when $\mu = 0$, we end up scoring with the regular relevance measure, whereas when $\mu = 1$, we score solely based on the marginal language model.

The reference model θ_O can be defined in several different ways, depending on how we summarize the set of reference topic models. We mention two representative possibilities here:

1. Using an average reference model

   We average all the reference models and treat all the reference models together as one single model θ_O, which is defined as:

   $$p(w \mid \theta_O) = \frac{1}{k-1} \sum_{i=1}^{k} p(w \mid \theta_i)$$

   The effect is to consider all the redundant parts in document d_k.

2. Using the closest reference model

   In this case, we use the reference model that is closest to the document model θ_k to represent the whole set of reference models. That is,

   $$\theta_O = \arg\min_{\theta_i : 1 \leq i \leq k-1} D(\theta_k || \theta_i)$$

   The effect is to consider only the most redundant part of document d_k. In this case, θ_O depends on d_k.

7.3.3 Evaluation

To evaluate the effectiveness of the proposed methods for combining novelty and relevance, we compare them with a well-tuned relevance-based ranking baseline. Since the marginal query model approach and the marginal document model approach can be expected to work well only when the query model is rich enough, we consider two relevance-ranking baselines. One, the simple baseline, is the best relevance-based ranking in terms of the aspect coverage measure using the original short queries. This simple baseline ranking is achieved when the Dirichlet prior smoothing parameter is set to 20,000. The other, the feedback baseline, uses an expanded (richer) query model based on this simple baseline. The expanded query model is an interpolation of the original maximum likelihood query model and a pseudo-feedback model with a coefficient of 0.5 on each model. The feedback model is estimated based on the top 100 documents (from the simple baseline results) using the mixture model approach described in Section 6.5.1 with the background noise parameter set to 0.5. The Dirichlet prior smoothing parameter is set to 5,000, which is optimal for scoring the expanded query.

We search for the best relevance-based ranking by trying many different values for the Dirichlet prior parameter. These results from using different smoothing parameter values are compared in Figure 7.3 to show how the relevance-based precision (i.e., average precision) and the aspect-based performance (i.e., the aspect coverage and uniqueness) respond to different values of the smoothing parameter.

The three pictures on the left side in Figure 7.3 show the comparison for the average, at recall=0.1, and at recall=0.5 for the simple query baseline. Overall, we see that when the Dirichlet prior parameter is set to a relatively large value, the curves are relatively flat, indicating that the sensitivity is low, especially for the aspect measures. However, in all cases, if the parameter value is too small, the performance can drop significantly. We also see that, while there is an overall correlation between the three curves, the optimal setting of the smoothing parameter is *different* for different measures. In particular, the optimal setting for relevance precision tends to be much lower than that for aspect measures. Indeed, the optimal setting for relevance precision is around 5,000, whereas it is generally above 10,000 for aspect measures. This seems to be true especially for the average performance and at recall of 0.5; for recall of 0.1, the optimal setting is relatively more consistent, which is also expected, as the performance is highly influenced by the rank of the very first, or first few relevant documents, thus the real difference between rankings may not be reflected. From these results, we use the ranking (at parameter value 20,000) that gives the

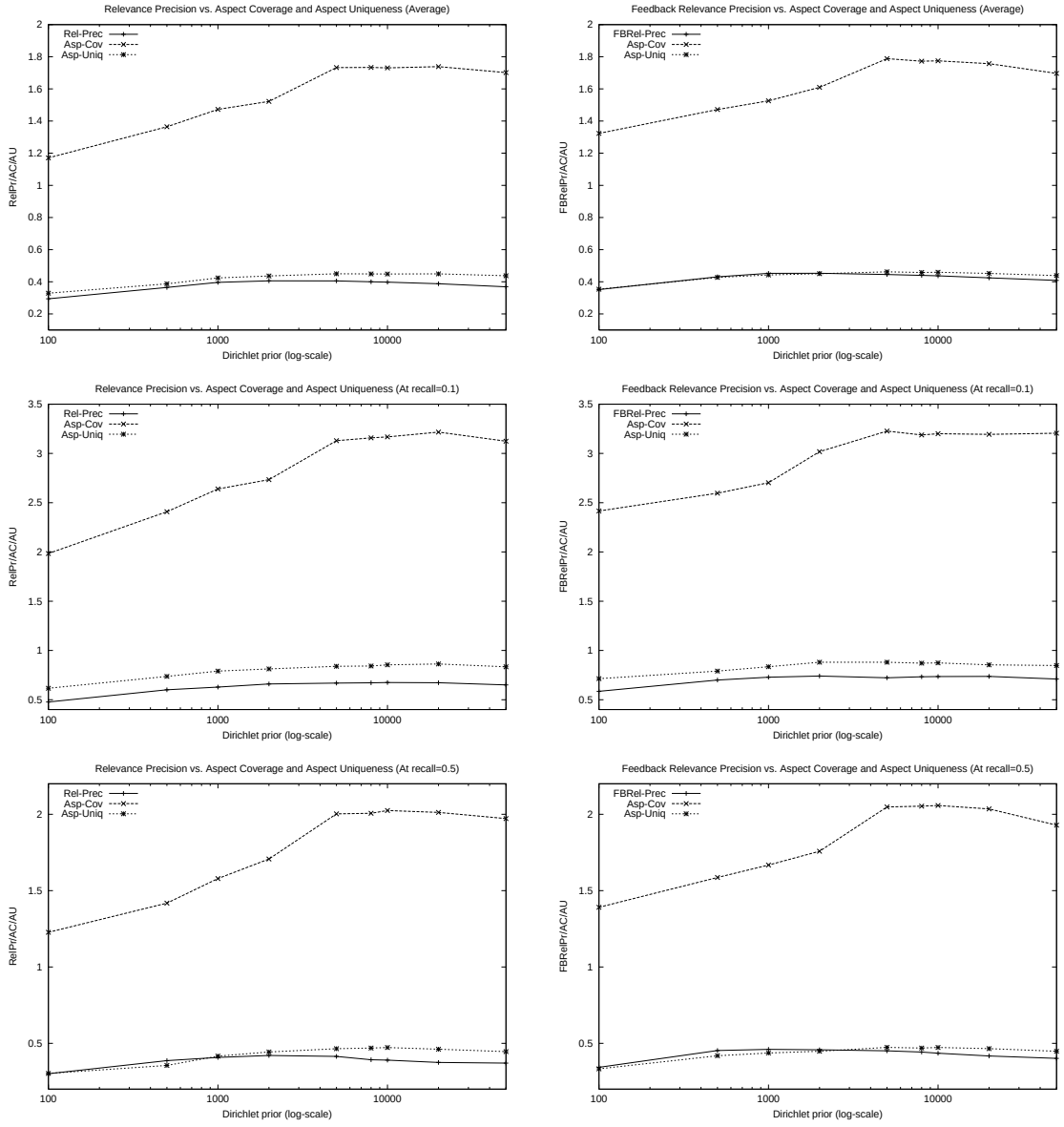

Figure 7.3: Correlation of relevance precision, aspect coverage, and aspect uniqueness as smoothing parameter changes when measured as average (top), at recall=0.1 (middle), and recall=0.5 (bottom). Left is simple queries; right is expanded queries. The performance is seen to be sensitive to the smoothing parameter, and the optimal setting of the smoothing parameter is different for each measure.

best aspect coverage as our simple relevance baseline.

On the right side of Figure 7.3, we show a similar comparison for the feedback query. We see that the overall patterns are similar to what we observed on the simple baseline results. The best average aspect coverage is achieved when the smoothing parameter is set

to 5,000.

The performance of the simple baseline and the feedback baseline is compared in Table 7.3. The feedback baseline improves over the simple baseline by all measures, though the amount of improvement for relevance-based measures is more significant. Thus, although pseudo feedback primarily aims at improving relevance-based performance, it also improves the aspect performance. Indeed, when ranking a mixture of relevant and non-relevant documents, improving the relevance-based precision will bring more relevant documents to the top of the list, which can be expected to improve the aspect coverage and aspect uniqueness, since both aspect measures would penalize having non-relevant documents on the top. Thus, in general, a better relevance-based average precision also means a better aspect coverage performance, suggesting that we could improve aspect coverage through improving relevance measurement. However, there is a limit in doing this; the best performing ranking according to the relevance measures does *not* also give the best aspect performance. Later, we will show that a combination of relevance measure and novelty measure can improve aspect performance even though it may have an inferior relevance-based performance.

Baseline	Rel. Precision			Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
Simple	0.414	0.708	0.404	1.737	3.216	2.008	0.449	0.863	0.460
Feedback	0.445	0.723	0.451	1.788	3.227	2.05	0.462	0.881	0.473
Improve	7.5%	2.1%	11.6%	2.9%	0.3%	2.1%	2.9%	2.1%	2.8%

Table 7.3: Comparison of simple baseline and feedback baseline.

With these two baselines, we can test the proposed methods for combining relevance and novelty. For each method, we examine the influence of the parameters and compare the method's performance with the baselines. For the sake of efficiency, the comparison is generally based on re-ranking some top-ranked documents according to the baseline ranking.

We first look at the cost-based combination of relevance and novelty. We use it for re-ranking the top 100 documents (for each topic) returned by the simple baseline relevance ranking as well as for re-ranking all the relevant documents. We vary the cost parameter ρ between 1 and 10. It is unreasonable to set ρ to any value below 1, as it would mean that a larger relevance value corresponds to a larger loss. As ρ becomes large, the combination relies more on relevance. When $\rho = 10$, it is almost like ranking based on only relevance.

The results are shown in Figure 7.4.

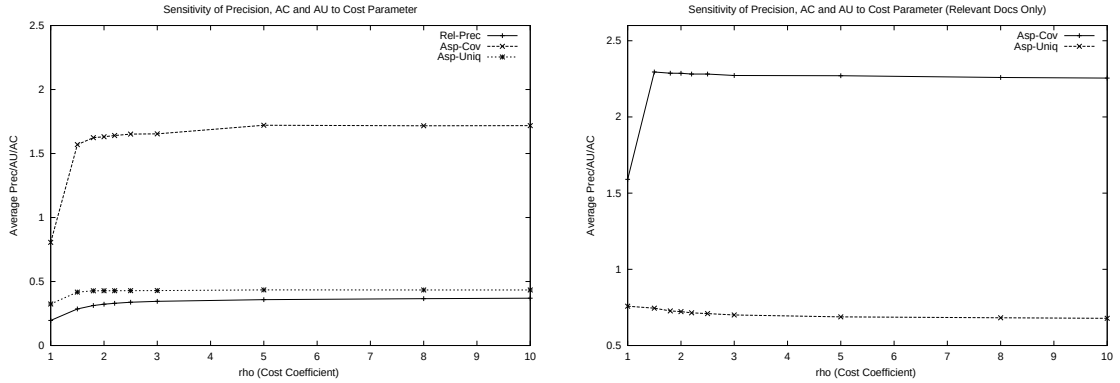

Figure 7.4: Effectiveness of the cost-based combination of relevance and novelty for re-ranking a mixture of relevant and non-relevant documents (left) and for re-ranking only relevant documents (right).

Ranking	Rel. Precision			Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
SimpBase	0.378	0.707	0.369	1.721	3.216	2.006	0.434	0.863	0.457
CC ($\rho = 5$)	0.358	0.656	0.354	1.721	3.210	1.970	0.434	0.868	0.442
Improve	-5.3%	-7.2%	-4.1%	0%	-0.2%	-1.8%	0%	+0.6%	-3.3%
FBBase	0.400	0.720	0.396	1.766	3.227	2.046	0.442	0.881	0.471
Improve	+5.8%	+1.8%	+7.3%	+2.6%	+0.3%	+2.0%	+1.8%	+2.1%	+3.1%

Ranking	Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
SimpBase	2.236	3.614	2.550	0.666	0.987	0.679
CC($\rho = 1.5$)	2.295	3.567	2.647	0.758	0.957	0.800
Improve	+2.6%	-1.3%	+3.8%	+13.8%	-3.0%	17.8%
FBBase	2.268	3.708	2.598	0.661	0.995	0.690
Improve	+1.4%	+2.6%	+1.9%	-0.8%	+0.8%	+1.6%

Table 7.4: Effectiveness of the cost-based combination of relevance and novelty for re-ranking a mixture of relevant and non-relevant documents (top) and for re-ranking relevant documents (bottom). The improvement of the feedback baseline over the simple baseline is also shown for comparison.

We see that the cost-based combination of relevance and novelty is not effective in improving either aspect coverage or aspect uniqueness when working on the mixed data (i.e., with both relevant and non-relevant documents). The feedback baseline results are obtained by using the expanded query to re-rank the same 100 documents. (The performance is thus worse than retrieving 1000 documents from the whole collection. Actually

it is also worse than retrieving the top 100 documents from the whole collection.) We see that the feedback baseline could improve over the simple baseline in terms of both relevance precision and aspect measures. It appears that in this testing collection, the novelty or redundancy is not a dominating factor compared with relevance; any improvement in measuring relevance can be expected to improve aspect coverage more significantly than trying to minimize redundancy. Indeed, even when ranking only the relevant documents, we see that feedback improves aspect coverage, though it hurts aspect uniqueness. This means that the improvement of relevance helps improve aspect coverage not only through "pushing down" non-relevant documents, but also through "bringing up" documents that tend to cover more aspects, and that is probably why feedback has a clear positive effect on aspect coverage when re-ranking all relevant documents.

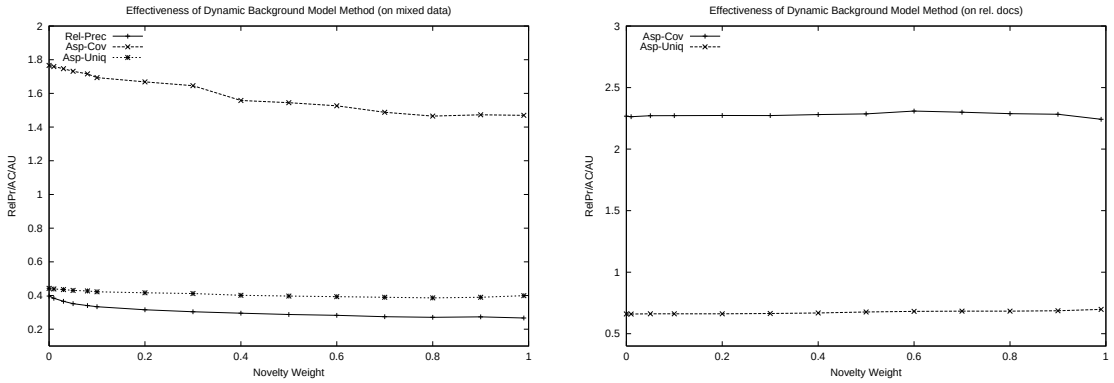

Figure 7.5: Effectiveness of the query background model approach to combining relevance and novelty for re-ranking mixed relevant and non-relevant documents (left) and for re-ranking only relevant documents (right).

We also see that the cost-based combination clearly and significantly improves the aspect uniqueness when re-ranking relevant documents, suggesting that it does help reduce the redundancy among documents, which, in turn, also helps improve the aspect coverage, as shown in Table 7.4.

Next, we evaluate the two dynamic query modeling approaches. First, we look at the query background model approach. The results on ranking both mixed documents (top 100 documents) and re-ranking only relevant documents are shown in Figure 7.5. We see that although aspect performance improves when re-ranking relevant documents, the performance is never improved when ranking mixed documents. A very likely reason is because the introduction of the query background helps non-relevant documents to "generate" the query and thus to get a higher score than they should get. Indeed, the relevance-based precision is clearly decreasing as we use a larger "novelty weight." What happens is that as

we use a higher novelty weight, more non-relevant documents are brought up, which hurts both the precision measure and aspect measures. Similar to the cost-based approach, when re-ranking relevant documents, the query background model approach is able to improve both aspect coverage and aspect uniqueness, as shown in Table 7.5.

Ranking	Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	2.268	3.708	2.598	0.661	0.995	0.690
QB ($\mu = 0.6$)	2.309	3.775	2.581	0.682	0.995	0.697
Improve	+1.8%	+1.8%	-0.7%	+3.2%	0.0%	+1.0%

Table 7.5: Effectiveness of the query background model on re-ranking relevant documents. It improves both aspect coverage and aspect uniqueness.

The results of the marginal query model method are shown in Figure 7.6 and Table 7.6. The results of the mixed data are based on re-ranking the top 30 documents returned by the simple query baseline. Overall, this method can slightly improve all the measures, but rather insignificantly, even when re-ranking relevant documents. It appears that as long as the novelty weight is not very high, the marginal query model approach does not change the ranking order significantly, but when the novelty weight is very high, the change of ordering hurts the performance of all the measures. These results suggest that the simple mixture model is not effective for estimating an accurate marginal query model.

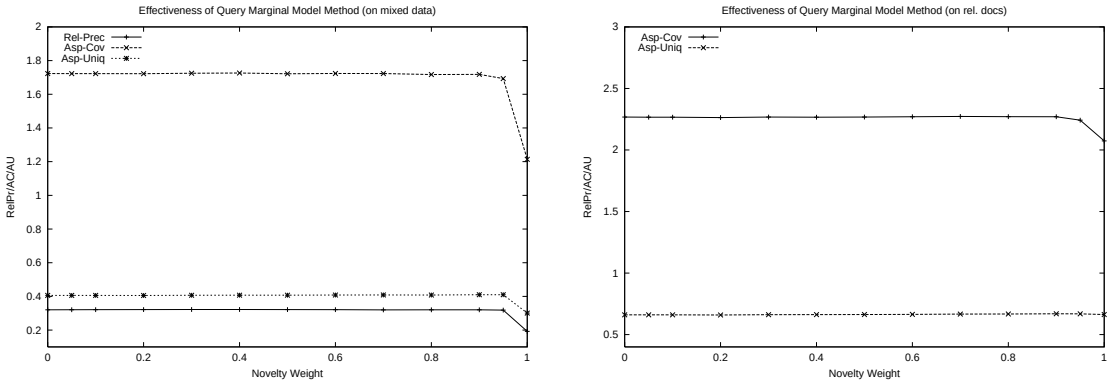

Figure 7.6: Effectiveness of the marginal query model approach to combining relevance and novelty for re-ranking mixed relevant and non-relevant documents (left) and for re-ranking only relevant documents (right).

The results of the marginal document model on re-ranking mixed data (top 30 documents) and re-ranking relevant documents are shown in Figure 7.7 and Table 7.7. We see

that this approach can actually improve all the measures on mixed documents, but quite insignificantly. It is interesting to note that this approach can hardly improve the performance when working on only relevant documents. This suggests that the improvement on the mixed data may be largely due to an improvement on relevance-ranking; indeed, there is a relatively significant increase in all the relevance-based measures. Although the mixture novelty measure is shown to be quite effective in improving the aspect uniqueness, the estimated marginal document model based on this novelty measure turns out not to be so effective.

Ranking	Rel. Precision			Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	0.320	0.707	0.214	1.723	3.223	1.991	0.406	0.877	0.423
MQM($\mu = 0.4$)	0.322	0.714	0.216	1.726	3.217	1.981	0.407	0.874	0.421
Improve	+0.6%	+1.0%	+0.9%	+0.2%	-0.2%	-0.5%	+0.2%	-0.3%	-0.5%

Ranking	Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	2.268	3.708	2.598	0.661	0.995	0.690
MQM($\mu = 0.1$)	2.273	3.708	2.590	0.666	0.995	0.691
Improve	+0.2%	0%	-0.3%	+0.8%	0%	+0.1%

Table 7.6: Effectiveness of the marginal query model approach to combining relevance and novelty for re-ranking a mixture of relevant and non-relevant documents (top) and for re-ranking relevant documents (bottom).

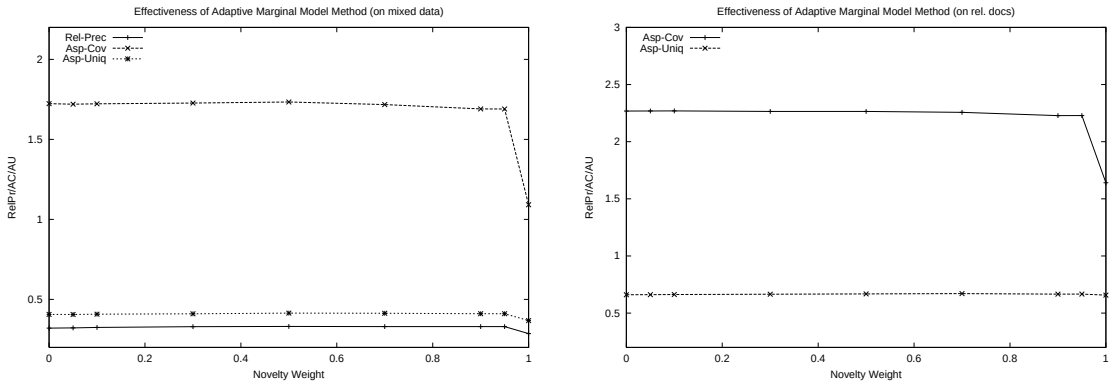

Figure 7.7: Effectiveness of the marginal document model approach to combining relevance and novelty for re-ranking mixed relevant and non-relevant documents (left) and for re-ranking only relevant documents (right).

In Table 7.8, we compare all four methods in terms of their best results on both the

Ranking Method	Rel. Precision			Aspect Coverage			Aspect Uniqueness		
	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	0.320	0.707	0.214	1.723	3.223	1.991	0.406	0.877	0.423
MDM($\mu = 0.5$)	0.331	0.722	0.238	1.734	3.283	1.922	0.415	0.898	0.425
Improve	+3.4%	+2.1%	+11.2%	+1.5%	+1.9%	-3.5%	+2.2%	+2.4%	+0.5%

Ranking Method	Aspect Coverage			Aspect Uniqueness		
	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	2.268	3.708	2.598	0.661	0.995	0.690
MDM($\mu = 0.1$)	2.269	3.708	2.580	0.662	0.995	0.688
Improve	0%	0%	-0.7%	+0.2%	0%	-0.3%

Table 7.7: Effectiveness of marginal document model approach to combining relevance and novelty for re-ranking a mixture of relevant and non-relevant documents (top) and for re-ranking relevant documents (bottom).

mixed data and the relevant data. On the mixed data, the marginal document model method and the marginal query model method could manage to improve both aspect coverage and aspect uniqueness, even though only slightly, but neither the cost-based combination method, nor the query background model method could improve over the relevance-ranking baseline. However, on the relevant data set, the results seem to be the opposite. The cost-based combination method and the query background model method perform better than the two marginal model approaches. Since we can factor out the relevance factor when re-ranking relevant documents, the improvement in aspect performance on the relevant data set is a good indicator of how well a method captures redundancy/novelty and how well the elimination of redundancy can help retrieve more relevant aspects. Thus, our results show that the cost-based combination method and the query background model method are both effective in removing redundancy and covering more aspects. However, neither approach can improve the aspect performance on the mixed data. This means that neither approach is very effective in ranking relevant documents above non-relevant ones; indeed, both approaches tend to decrease the relevance-based precision significantly (see Table 7.4). As a result, even though the "internal" ranking of relevant documents may have been improved, the overall performance decreases because too many non-relevant documents are ranked high. Clearly, to perform well on a mixed data set, a method must be able to model relevance and redundancy effectively. The cost-based method and the query background model method are examples where the modeling of redundancy is effective, but the modeling of relevance is relatively weak. On the other hand, the two marginal model approaches appear to be better at modeling relevance than redundancy. Indeed, their improvement on the rel-

evant data set is quite small, but both improve the performance slightly on the mixed data set. From Table 7.6 and Table 7.7, we see that these two approaches both also improve the precision on the mixed data set. Thus, the improvement in aspect performance is mainly due to an improvement of relevance ranking. This is an interesting observation, as neither of them were designed to improve relevance ranking. One possible explanation for the marginal query model method is that the dynamic query model is effectively a "distilled" query model with the background noise excluded; this strategy has been shown to be effective in improving precisions (see Chapter 6, Section 6.4). However, more study is needed to understand why the marginal document model method improves precision.

Since the mixture model novelty measure has already been shown to be effective in measuring novelty/redundancy, it is not very surprising that the cost-based combination method, which uses the mixture model novelty measure, improves the aspect uniqueness significantly on the relevant data set. However, what is interesting is that using the mixture model novelty measure has not been able to improve the aspect coverage, but combining it with relevance makes it possible to improve both aspect uniqueness and aspect coverage.

In summary, on the relevant data set, the cost-based combination method is most effective and it can improve both aspect coverage and aspect uniqueness, suggesting that eliminating redundancy does help improve aspect coverage. However, on the mixed data set, we have not yet seen a case where the relevance-based precision is decreased, but the aspect performance is improved; instead, the improvement in aspect performance is often also accompanied by an increase in relevance-based performance. Thus, it appears that improving aspect coverage through eliminating redundancy is effective only when we can achieve/ensure a very good relevance ranking; when the relevance ranking is not very accurate, improving the relevance ranking may be expected to have more impact on the aspect performance than removing redundancy. Since whether eliminating redundancy can improve aspect coverage highly depends on how much redundancy is in our data, these observations suggest that the percentage of redundant documents may be relatively low when compared with the percentage of non-relevant documents in our data set. Therefore, it would be very useful to test all these methods with some synthetic data where we can control the level of redundancy.

In practice, for such an aspect retrieval task, we may consider using either the cost-based combination method ($\rho = 1.5$) or the marginal document model ($\mu = 0.5$), depending on how accurate our relevance measure is. The cost-based combination method would be appropriate if we can expect to achieve a very high precision; otherwise, the marginal

Ranking Method	Aspect Coverage Increase			Aspect Uniqueness Increase		
	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
CC(ρ)	0% ($+\infty$)	0%($+\infty$)	0%($+\infty$)	0%($+\infty$)	0%($+\infty$)	0%($+\infty$)
QB (μ)	0% (0)	0%(0)	0%(0)	0%(0)	0% (0)	0%(0)
MQM (μ)	+0.2%(0.4)	-0.2%(0.4)	-0.5%(0.4)	+1.0%(0.95)	+1.7%(0.95)	-1.6% (0.95)
MDM (μ)	+1.5%(0.5)	+1.9%(0.5)	-3.5%(0.5)	+2.2%(0.5)	+2.4% (0.5)	+0.5% (0.5)

Ranking Method	Aspect Coverage Increase			Aspect Uniqueness Increase		
	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
CC(ρ)	+2.6% (1.5)	-1.3%(1.5)	+3.8%(1.5)	+13.8% (1.5)	-3.0%(1.5)	+17.8% (1.5)
QB (μ)	+1.8%(0.6)	+1.8%(0.6)	-0.7% (0.6)	+5.6% (0.99)	+0.5%(0.99)	+3.0% (0.99)
MQM (μ)	+0.2% (0.1)	0%(0.1)	-0.3%(0.1)	+1.2% (0.9)	+0.1% (0.9)	+0.3% (0.9)
MDM (μ)	0%(0.1)	0%(0)	0%(0)	+1.1%(0.5)	+0.1%(0.5)	-0.1%(0.5)

Table 7.8: Comparison of all MMR models for re-ranking mixed data (top) and for re-ranking relevant documents (bottom). The numbers shown are the increase over the baseline relevance ranking.

document model method may be more appropriate.

7.4 Maximal Diverse Relevance (MDR) Models

In this section, we study another completely different type of models for the aspect retrieval task. Unlike in the previous section, where we hope to increase aspect coverage *indirectly* through eliminating the redundancy among documents, here we intend to improve the aspect coverage more *directly* through modeling the possible aspects in the documents with mixture language models.

7.4.1 A General Aspect Retrieval Model

To model the aspect retrieval problem, we consider the generative model illustrated in Figure 7.8. We assume that there is a space of A aspects, each characterized by a unigram language model. Formally, let $\boldsymbol{\tau} = (\tau_1, ..., \tau_A)$ be a vector of aspects. τ_i is a unigram language model and $p(w \mid \tau_i)$ gives the probability of word w according to the aspect τ_i.

Now, let us assume that a user, with an interest in retrieving documents to cover some of these A aspects, would first pick a probability distribution θ_Q over the aspects, and then formulate a query according to a query generation model $p(\mathbf{q} \mid \boldsymbol{\tau}, \theta_Q)$. Intuitively, θ_Q encodes

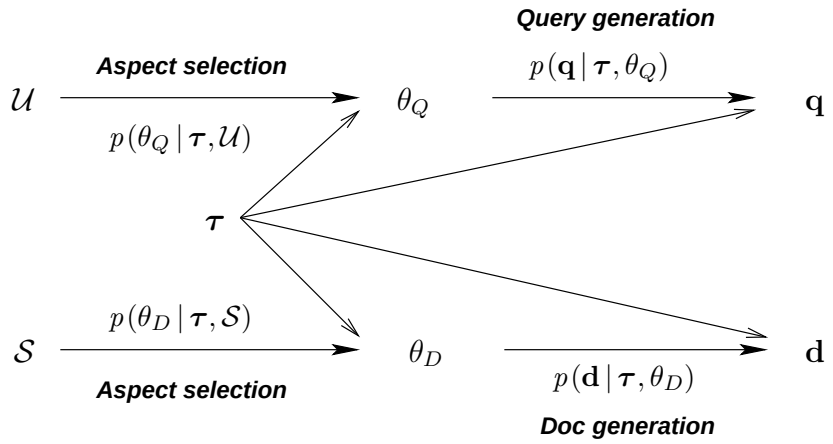

Figure 7.8: Aspect generative model of query $\mathbf{q}$ and document $\mathbf{d}$.

the user's preferences on aspect coverage, and in general, would have the probability mass concentrated on those aspects that are most interesting to the user; other non-interesting aspects may have a zero probability. Furthermore, among these "interesting aspects," the distribution is generally non-uniform, reflecting the fact that some aspects are more emphasized than others. Similarly, we also assume that the author or source of a document $\mathbf{d}$ would first pick an aspect coverage distribution θ_D, and then generate $\mathbf{d}$ according to a document generation model $p(\mathbf{d} \mid \boldsymbol{\tau}, \theta_D)$.

For example, a simple case of $p(\mathbf{d} \mid \boldsymbol{\tau}, \theta_D)$ would be a mixture model, in which θ_D is the mixing weights and $\boldsymbol{\tau}$ are the component unigram language models. That is, suppose $\mathbf{d} = d_1 d_2 ... d_n$, we have

$$p(\mathbf{d} \mid \boldsymbol{\tau}, \theta_D) = \prod_{i=1}^{n} \sum_{j=1}^{A} p(j \mid \theta_D) p(d_i \mid \tau_j)$$

But the derivation below is not restricted to such a mixture model.

To derive an aspect retrieval model, we start with the following general greedy algorithm ranking formula:

$$r(\mathbf{d}_k | \mathbf{d}_1, ..., \mathbf{d}_{k-1}, \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) \stackrel{\text{def}}{=} \int_{\Theta} l(\mathbf{d}_k | \mathbf{d}_1, ..., \mathbf{d}_{k-1}, \theta, F(\mathcal{U}), F(\vec{\mathcal{S}})) p(\theta | \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) d\theta$$

This conditional risk gives us a way to evaluate the rest of the documents and pick the best d_k, given that we have already picked $\mathbf{d}_1$, ..., $\mathbf{d}_{k-1}$. With the generative models given above, $\theta = (\boldsymbol{\tau}, \theta_Q, \theta_{D_1}, ..., \theta_{D_k})$.

We now consider the following loss function:

$$\begin{aligned} l(\mathbf{d}_k|\mathbf{d}_1,...,\mathbf{d}_{k-1},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}})) &= l(\mathbf{d}_k|\mathbf{d}_1,...,\mathbf{d}_{k-1},\boldsymbol{\tau},\theta_Q,\theta_{D_1},...,\theta_{D_k}) \\ &= D(\theta_Q||\theta^{D_k}_{D_1...D_{k-1}}) \end{aligned}$$

where $\theta^{D_k}_{D_1...D_{k-1}}$ is a weighted average of $\{\theta_{d_i}\}_{i=1}^{k}$ defined as follows:

$$p(a\,|\,\theta^{D_k}_{D_1...D_{k-1}}) = \frac{\mu}{k-1}\sum_{i=1}^{k-1} p(a\,|\,\theta_{D_i}) + (1-\mu)p(a\,|\,\theta_{D_k})$$

where $\mu \in (0,1]$ is a parameter indicating how much redundancy we would like to model.

The idea behind this loss function is the following: We expect θ_Q to give us a "measure" of which aspect is relevant – a high $p(a\,|\,\theta_Q)$ indicates that the aspect a is likely a relevant one. The loss function encodes our preferences for a similar "aspect coverage distribution" given by all the documents $d_1,...,d_k$. Thus, if θ_Q assigns high probabilities to some aspects, then we would expect to cover these (presumably relevant) aspects more than other aspects. The best d_k is thus the one that can work together with $d_1,...,d_{k-1}$ to achieve a coverage distribution that is most similar to the desired aspect coverage based on the query, i.e., $p(a\,|\,\theta_Q)$. The parameter μ controls how much we rely on the previously picked documents $d_1,...,d_{k-1}$ to cover the aspects. If we do not rely on them (i.e., $\mu = 0$), we will be looking for a d_k that best covers all the relevant aspects by itself. On the other hand, if $\mu > 0$, part of the coverage would have been explained by the previously picked documents, and the best d_k would be one that best covers those "under-covered" relevant aspects. Essentially, we are searching for the $\mathbf{d}_k$ that best supplements the coverage provided by the previously picked documents with respect to the desired coverage θ_Q.

Putting this loss function and the aspect generative model into the conditional risk formula, we have

$$\begin{aligned} r(\mathbf{d}_k|\mathbf{d}_1,...,\mathbf{d}_{k-1},\mathbf{q},\mathcal{C},\mathcal{U},\vec{\mathcal{S}}) &= \int_\Theta l(\mathbf{d}_k|\mathbf{d}_1,...,\mathbf{d}_{k-1},\theta,F(\mathcal{U}),F(\vec{\mathcal{S}}))p(\theta|\mathbf{q},\mathcal{C},\mathcal{C}_k,\mathcal{U},\vec{\mathcal{S}})d\theta \\ &= \int_\Theta D(\theta_Q||\theta^{D_k}_{D_1...D_{k-1}})p(\theta_Q,\theta_{D_1},...,\theta_{D_k}\,|\,\mathbf{q},\mathcal{C},\mathcal{C}_k,\mathcal{U},\vec{\mathcal{S}})d\theta_Q d\theta_{D_1}...d\theta_{D_k} \end{aligned}$$

and

$$\begin{aligned} p(\theta_Q, \theta_{D_1}, ..., \theta_{D_k} \mid \mathbf{q}, \mathcal{C}, \mathcal{C}_k, \mathcal{U}, \vec{\mathcal{S}}) &= \int_{\mathcal{T}} p(\theta_Q, \theta_{D_1}, ..., \theta_{D_k}, \boldsymbol{\tau} \mid \mathbf{q}, \mathcal{C}, \mathcal{C}_k, \mathcal{U}, \vec{\mathcal{S}}) d\boldsymbol{\tau} \\ &= \int_{\mathcal{T}} p(\theta_Q \mid \boldsymbol{\tau}, \mathbf{q}, \mathcal{U}) \prod_{i=1}^{k} p(\theta_{D_i} \mid \boldsymbol{\tau}, \mathbf{d}_i, \vec{\mathcal{S}}) p(\boldsymbol{\tau} \mid \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) d\boldsymbol{\tau} \\ &\approx p(\theta_Q \mid \hat{\boldsymbol{\tau}}, \mathbf{q}, \mathcal{U}) \prod_{i=1}^{k} p(\theta_{D_i} \mid \hat{\boldsymbol{\tau}}, \mathbf{d}_i, \vec{\mathcal{S}}) p(\hat{\boldsymbol{\tau}} \mid \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) \end{aligned}$$

where $\hat{\boldsymbol{\tau}} = \arg\max_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}})$, and $\mathcal{C}_k = \{\mathbf{d}_1, ..., \mathbf{d}_k\}$.

Note that we have assumed that $\boldsymbol{\tau}$ can be estimated using all the documents in the collection, i.e., $\mathcal{C}$, so $p(\boldsymbol{\tau} \mid \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}})$ does not depend on d_k and can be ignored for the purpose of ranking d_k. That is,[3]

$$\begin{aligned} r(\mathbf{d}_k|\mathbf{d}_1, ..., \mathbf{d}_{k-1}, \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) &\overset{\text{rank}}{\approx} \int_{\Theta} D(\theta_Q || \theta^{D_k}_{D_1 ... D_{k-1}}) p(\theta_Q \mid \hat{\boldsymbol{\tau}}, \mathbf{q}, \mathcal{U}) \prod_{i=1}^{k} p(\theta_{D_i} \mid \hat{\boldsymbol{\tau}}, \mathbf{d}_i, \vec{\mathcal{S}}) \\ &\overset{\text{rank}}{\approx} D(\widehat{\theta}_Q || \widehat{\theta}^{D_k}_{D_1 ... D_{k-1}}) p(\widehat{\theta}_Q \mid \hat{\boldsymbol{\tau}}, \mathbf{q}, \mathcal{U}) \prod_{i=1}^{k} p(\widehat{\theta}_{D_i} \mid \hat{\boldsymbol{\tau}}, \mathbf{d}_i, \vec{\mathcal{S}}) \\ &\overset{\text{rank}}{=} D(\widehat{\theta}_Q || \widehat{\theta}^{D_k}_{D_1 ... D_{k-1}}) p(\widehat{\theta}_{D_k} \mid \hat{\boldsymbol{\tau}}, \mathbf{d}_k, \vec{\mathcal{S}}) \\ &\overset{\text{rank}}{\approx} D(\widehat{\theta}_Q || \widehat{\theta}^{D_k}_{D_1 ... D_{k-1}}) \end{aligned}$$

where

$$\widehat{\theta}_Q = \arg\max_{\theta_Q} p(\theta_Q \mid \hat{\boldsymbol{\tau}}, \mathbf{q}, \mathcal{U})$$

and

$$\widehat{\theta}_{D_i} = \arg\max_{\theta_{D_i}} p(\theta_{D_i} \mid \hat{\boldsymbol{\tau}}, \mathbf{d}_i, \vec{\mathcal{S}})$$

Thus, we have obtained the following ranking procedure:

1. We rank all the documents in a greedy fashion, using the conditional risk $r(\mathbf{d}_k|\mathbf{d}_1, ..., \mathbf{d}_{k-1}, \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}})$ when selecting the k-th document.
2. Before selecting any document, we first estimate $\boldsymbol{\tau}$, i.e., $\hat{\boldsymbol{\tau}} = \arg\max_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}})$.
3. To compute $r(\mathbf{d}_k|\mathbf{d}_1, ..., \mathbf{d}_{k-1}, \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}})$, we first compute $\widehat{\theta}_Q$ and $\widehat{\theta}_{D_k}$ (presumably, $\widehat{\theta}_{D_1}, ..., \widehat{\theta}_{D_{k-1}}$ were already computed), and then evaluate $D(\widehat{\theta}_Q || \widehat{\theta}^{D_k}_{D_1 ... D_{k-1}})$. $\widehat{\theta}_Q$ and $\widehat{\theta}_{D_k}$ are computed using the formulas given above.

[3] In our experiments with re-ranking a pre-selected working set of documents, we estimate $\boldsymbol{\tau}$ based on the working set.

In order to make this general aspect retrieval model operational, we need to specify:

- Query model: $p(\mathbf{q} \mid \boldsymbol{\tau}, \theta_Q)$ and $p(\theta_Q \mid \boldsymbol{\tau}, \mathcal{U})$
- Document model: $p(\mathbf{d} \mid \boldsymbol{\tau}, \theta_D)$ and $p(\theta_D \mid \boldsymbol{\tau}, \vec{\mathcal{S}})$

If we have one distinct aspect τ_w for each word w and τ_w is entirely concentrated on the single word w, i.e., $p(w \mid \tau_w) = 1$ and for $w' \neq w$, $p(w' \mid \tau_w) = 0$, then θ_Q and θ_D are just regular unigram language models. In this case, if we set $\lambda = 1$, the conditional risk would no longer depend on the previously picked documents, and is given by

$$r(\mathbf{d}_k|\mathbf{d}_1, ..., \mathbf{d}_{k-1}, \mathbf{q}, \mathcal{C}, \mathcal{U}, \vec{\mathcal{S}}) = \int_{\Theta} D(\theta_Q||\theta_{D_k}) p(\theta_Q \mid \mathbf{q}, \mathcal{U}) p(\theta_{D_k} \mid \mathbf{d}_k, \vec{\mathcal{S}}) d\theta_Q d\theta_{D_k}$$

This is precisely the regular ranking formula with *independent* KL-divergence loss function. Thus, the aspect retrieval model is a generalization of the regular KL-divergence retrieval model.

When we choose a $\lambda < 1$, we achieve an effect of "aspect-covering" ranking, since when picking $\mathbf{d}_k$, we will be concentrating on the under-covered aspects.

In general, we can plug in any specific aspect-based generative models to the general aspect retrieval model, leading to potentially different retrieval formulas. We now discuss some specific aspect-based generative models.

7.4.2 Query Model $p(\mathbf{q} \mid \boldsymbol{\tau}, \theta_Q)$ and $p(\theta_Q \mid \boldsymbol{\tau}, \mathcal{U})$

Let $\mathbf{q} = q_1 ... q_m$ be a query. The query generation model can be the following mixture model:

$$p(\mathbf{q} \mid \boldsymbol{\tau}, \theta_Q) = \prod_{i=1}^{m} \sum_{a=1}^{A} p(a \mid \theta_Q) p(q_i \mid \tau_a)$$

The coverage selection model $p(\theta_Q \mid \boldsymbol{\tau}, \mathcal{U})$ should reflect our prior on user's preferences on aspect coverage. The simplest case is to use a non-informative uniform prior, but it would be more interesting to consider using a Dirichlet distribution with parameters $\boldsymbol{\alpha}_Q(\boldsymbol{\tau})$, i.e., $\mathcal{D}(\theta_Q \mid \boldsymbol{\alpha}_Q(\boldsymbol{\tau}))$. The parameters $\boldsymbol{\alpha}_Q(\boldsymbol{\tau})$ depend on $\boldsymbol{\tau}$, and intuitively reflect the popularity of the aspects. With the Dirichlet prior, the user would first sample a desired aspect coverage distribution θ_Q using the Dirichlet coverage selection model $\mathcal{D}(\theta_Q \mid \boldsymbol{\alpha}_Q(\boldsymbol{\tau}))$, and then generate the query $\mathbf{q}$ according to the mixture model that uses θ_Q as the mixing weights. Note that we assume that $\boldsymbol{\tau}$ are given in advance.

7.4.3 Document Model $p(\mathbf{d} \mid \boldsymbol{\tau}, \theta_D)$ and $p(\theta_D \mid \boldsymbol{\tau}, \vec{\mathcal{S}})$

Let $\mathbf{d} = d_1 ... d_n$ be a document. We first mention the plain mixture model, where we assume that all the words in a document are generated using an uncertain, but fixed aspect. That is,

$$p(\mathbf{d} \mid \boldsymbol{\tau}, \theta_D) = \sum_{a=1}^{A} p(a \mid \theta_D) \prod_{i=1}^{n} p(d_i \mid \tau_a)$$

However, this model does not reflect our goal of capturing multiple aspects within one single document, so we do not further explore this model.

In order to model multiple aspects in one document, we use the following aspect-based generative mixture model:

$$p(\mathbf{d} \mid \boldsymbol{\tau}, \theta_D) = \prod_{i=1}^{n} \sum_{a=1}^{A} p(a \mid \theta_D) p(d_i \mid \tau_a)$$

We assume that an imaginary author (for some document source $\mathcal{S}$) would first pick a desired aspect coverage distribution according to $p(\theta_D \mid \boldsymbol{\tau}, \mathcal{S})$, and then generate the document $\mathbf{d}$ using this mixture model in which θ_D is used as mixing weights on the aspects.

Now, the question is how to define the coverage selection probability $p(\theta_D \mid \boldsymbol{\tau}, \mathcal{S})$. It turns out that with two different choices of $p(\theta_D \mid \boldsymbol{\tau}, \mathcal{S})$, we end up with two well-known aspect-based models – probabilistic Latent Semantic Indexing (Hofmann, 1999) and Latent Dirichlet Allocation (Blei et al., 2003).

Probabilistic Latent Semantic Indexing (PLSI)

Suppose we let $p(\theta_D \mid \boldsymbol{\tau}, \mathcal{S})$ be defined only on the set of documents in our collection $\mathcal{C} = \{d_1, ..., d_N\}$. That is, we only allow N different coverage distributions, or

$$\sum_{i=1}^{N} p(\theta_{D_i} \mid \mathcal{S}) = 1$$

And suppose we do not impose any other constraints.

Under this assumption, the log-likelihood of the whole collection is

$$l(\boldsymbol{\tau}, \{\theta_{D_i}\}_{i=1}^{N} | \mathcal{C}) = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \log(\sum_{a=1}^{A} p(a \mid \theta_{D_i}) p(d_{ij} \mid \tau_a))$$

where n_i is the length of document d_i. The parameters for this model include the subtopic language models $\boldsymbol{\tau}$ and a potentially different mixing weight distribution θ_{D_i} for each document d_i. Thus, this is essentially the Probabilistic Latent Semantic Indexing (PLSI) proposed in (Hofmann, 1999).

To apply this model to the problem of aspect retrieval, we first estimate $\boldsymbol{\tau}$ and θ_{D_i}, e.g., by maximizing the collection likelihood given above. The EM algorithm can be applied. See (Hofmann, 1999) for more discussion of the estimation methods and how to avoid over-fitting.

Suppose $\hat{\boldsymbol{\tau}}$ and $\hat{\theta}_{D_i}$ are the estimated models. Now, we can compute $\hat{\theta}_Q$ by maximizing the posterior probability, $p(\theta_Q \mid \hat{\boldsymbol{\tau}}, \mathbf{q}, \mathcal{U})$, i.e.,

$$\hat{\theta}_Q = \arg\max_{\theta_Q} p(\mathbf{q} \mid \hat{\boldsymbol{\tau}}, \theta_Q) p(\theta_Q \mid \boldsymbol{\tau}, \mathcal{U})$$

This is a MAP optimization problem, when we use the Dirichlet distribution $D(\theta_Q \mid \boldsymbol{\alpha}_Q(\hat{\boldsymbol{\tau}}))$ for the aspect selection model $p(\theta_Q \mid \boldsymbol{\tau}, \mathcal{U})$. It can be solved using the EM algorithm. The updating equations are:

$$p^{(n+1)}(a \mid \theta_Q) = \frac{\sum_{j=1}^{m} p(a \mid q_i) + \boldsymbol{\alpha}_{Q,a}}{\sum_{a'=1}^{A}(\sum_{j=1}^{m} p(a' \mid q_i) + \boldsymbol{\alpha}_{Q,a'})}$$

where

$$p(a \mid q_i) = \frac{p^{(n)}(a \mid \theta_Q) p(q_i \mid \hat{\tau}_a)}{\sum_{a'=1}^{A} p^{(n)}(a' \mid \theta_Q) p(q_i \mid \hat{\tau}'_a)}$$

and $\boldsymbol{\alpha}_{Q,a}$ is the component value of $\boldsymbol{\alpha}_Q$ corresponding to aspect a.

Finally, we can use $D(\hat{\theta}_Q || \hat{\theta}^{D_k}_{D_1 \ldots D_{k-1}})$ to score documents.

One problem with the PLSI is the large number of parameters. Indeed, the total number of parameters is $A(|V| + N)$, where $|V|$ is the vocabulary size, A is the number of aspects, and N is the number of documents. This is a serious problem, especially for modeling aspects, since the number of parameters grows linearly according to both the number of aspects and the number of documents. Next, we discuss the Latent Dirichlet Allocation model, which effectively uses a Dirichlet distribution to "regulate" the θ_D for all the documents, reducing the number of parameters significantly.

Latent Dirichlet Allocation (LDA)

Instead of considering only a *finite* set of coverage distributions, let us consider the continuous space of all the possible aspect coverage distributions, which is the space of all

the multinomial distributions over the aspects. And we let $p(\theta_D \mid \boldsymbol{\tau}, \mathcal{S})$ be Dirichlet with parameter $\boldsymbol{\alpha}_D$, i.e., $\mathcal{D}(\theta_D \mid \boldsymbol{\alpha}_D)$.

Under this assumption, the log-likelihood of the whole collection is

$$l(\boldsymbol{\tau}, \boldsymbol{\alpha}_D|\mathcal{C}) = \sum_{i=1}^{N} \log \int_{\Theta_D} \mathcal{D}(\theta_D \mid \boldsymbol{\alpha}_D) \prod_{j=1}^{n_i} \sum_{a=1}^{A} p(a \mid \theta_D) p(d_{ij} \mid \tau_a) d\theta_D$$

This is precisely the LDA model proposed in (Blei et al., 2003).

The LDA model has significantly fewer parameters, including only the aspect language models $\boldsymbol{\tau}$ and the Dirichlet parameters $\boldsymbol{\alpha}_D$. In order to obtain the aspect coverage distribution θ_D for each document, we can compute the posterior distribution of θ_D and use the mode as an estimate.

In principle, the parameters of the LDA model can also be estimated using a maximum likelihood estimator by maximizing the collection likelihood given above. In practice, however, the estimation of parameters for the LDA model is significantly more complicated than for the PLSI model. A variational approach has been used in (Blei et al., 2003), and an Expectation-Propagation algorithm is proposed and studied in (Minka and Lafferty, 2002). A comparison and analysis of both approaches is also given in (Minka and Lafferty, 2002).

Once we have obtained an estimate for the parameters (i.e., $\hat{\boldsymbol{\tau}}$ and $\hat{\boldsymbol{\alpha}}_D$) through either of these two algorithms, we can then estimate the query aspect coverage distribution θ_Q in exactly the same way as we have done for the PLSI model, i.e.,

$$\hat{\theta}_Q = \arg\max_{\theta_Q} p(\mathbf{q} \mid \hat{\boldsymbol{\tau}}, \theta_Q) p(\theta_Q \mid \hat{\boldsymbol{\tau}}, \mathcal{U})$$

Similarly, to estimate the aspect coverage distribution of a document $\mathbf{d} = d1...d_n$, we can use

$$\begin{aligned} \hat{\theta}_D &= \arg\max_{\theta_D} p(\mathbf{d} \mid \hat{\boldsymbol{\tau}}, \theta_D) \mathcal{D}(\theta_D \mid \hat{\boldsymbol{\alpha}}_D) \\ &= \arg\max_{\theta_D} \mathcal{D}(\theta_D \mid \hat{\boldsymbol{\alpha}}_D) \prod_{j=1}^{n} \sum_{a=1}^{A} p(a \mid \theta_D) p(d_j \mid \hat{\tau}_a) \end{aligned}$$

This is again a MAP estimation problem, for which we can use the EM algorithm.

Once we have $\hat{\theta}_Q$ and $\hat{\theta}_D$ for all the candidate documents, we can score documents based on $D(\hat{\theta}_Q||\hat{\theta}^{D_k}_{D_1...D_{k-1}})$.

7.4.4 Evaluation

Probabilistic LSI

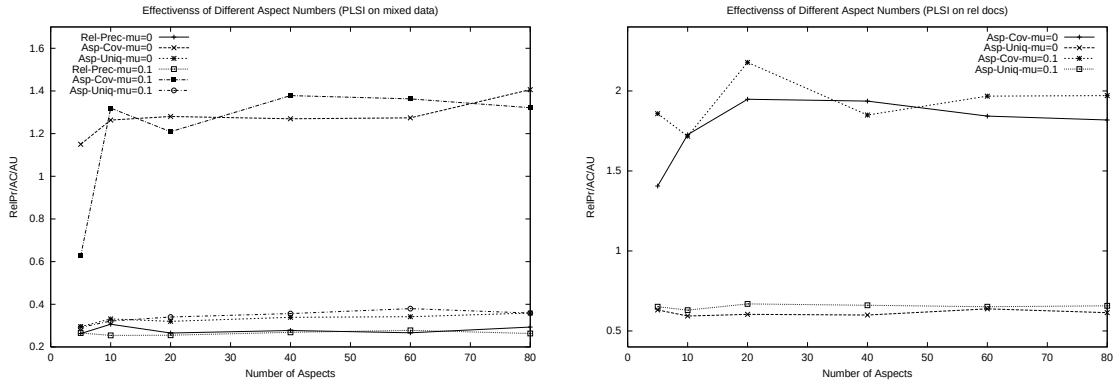

Figure 7.9: Influence of the number of aspects in the PLSI aspect model on mixed data (left) and relevant data (right).

All the PLSI experiments are based on re-ranking the top 30 documents returned from a slightly different relevance ranking baseline. [4] This baseline has been obtained by the same feedback procedure, but with slightly different (and inferior) parameter setting. This makes the PLSI results not comparable with other results. In the future, we want to conduct more controlled experiments to compare these different methods.

We first look at the influence of the choice of the number of aspects in the PLSI model. The number of aspects determines the granularity of the aspects under consideration. Assuming a small number of aspects would mean relatively general aspects, from the viewpoint of discriminating aspects, it is presumably better to consider a larger number of aspects. However, parameter estimation would be more reliable if we only consider a small number of aspects. Thus, empirically, there would be a trade-off involved when determining the optimal number of aspects.

The performance using a different number of aspects is shown in Figure 7.9. In each plot, we show both the aspect coverage and the aspect uniqueness values, and in the case of re-ranking the mixed data, we also show the average precision. We show the performance for both $\mu = 0$ and $\mu = 0.1$. From these results, we see that the performance is sensitive to the choice of the number of clusters. However, it is hard to see any pattern. We do see that using $\mu = 0.1$ improves the aspect uniqueness over using $\mu = 0$. This is expected since

[4] The PLSI experiments were performed before we established the baselines for all other methods.

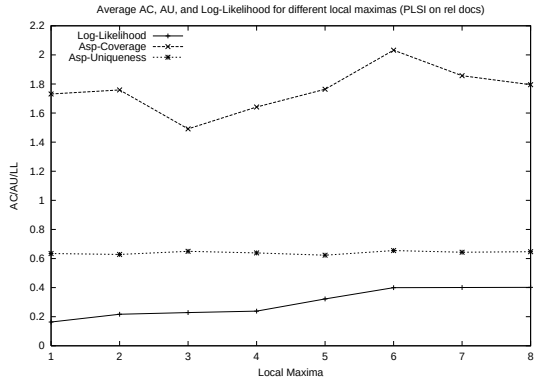

Figure 7.10: Comparison of the average aspect performance and likelihood among 8 different local maximas.

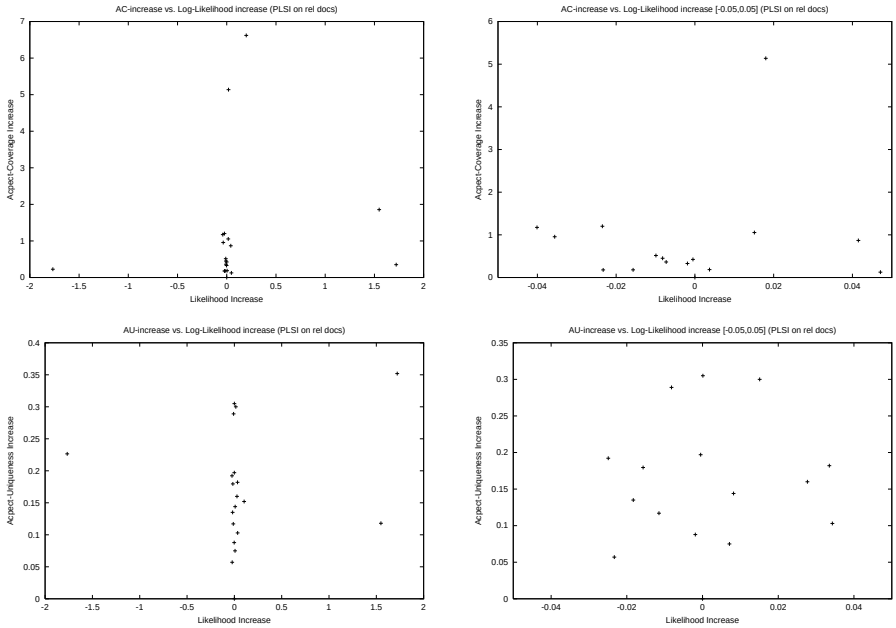

Figure 7.11: Comparison of aspect performance and likelihood between the best local maxima and the worst maxima. Top is aspect coverage; bottom is aspect uniqueness. Left has all the data points (i.e., queries); right has only the data points with a log-likelihood within the range [-0.05,0.05].

μ indicates how much redundancy we want to consider. Working with a relatively large number of aspects appears to help improve the aspect uniqueness, and sometimes also the aspect coverage, but it is not consistent.

To avoid being trapped at a local maxima, our experiment program automatically tries ten different initial values for the EM algorithm, and uses the one that gives the highest

likelihood. However, since the PLSI model has many local maximas, it is still very likely that the best local maxima out of the ten trials would not be the global maxima. This means that the sensitivity to the number of clusters can be caused by the variances in converging to different local maximas. To clarify this, we repeat our experiments eight times and let the EM algorithm converge to eight potentially different local maximas. We first determine if a higher likelihood (i.e., a better local maxima) leads to a better performance. In Figure 7.10, we compare the *average* aspect performance and the likelihood among these different local maximas. It seems that there is some (rather weak) correlation between the average log-likelihood and the aspect coverage. The correlation between the likelihood and the aspect uniqueness is completely unclear. Next, for each query, we identify the best-performing and worst-performing local maximas, and we compute the performance difference and the likelihood difference between the best-performing local maxima and the worst-performing local maxima. In Figure 7.11, we plot the performance difference and the likelihood difference for each query. We see that the performance is extremely sensitive to the local maxima, and there is no obvious correlation between the increase of aspect performance and the increase of likelihood. That is, a local maxima with a higher likelihood does not necessarily lead to a better aspect performance. Of course, it is possible that none of these local maximas is actually close to the global maxima. All these results suggest that it is very important to find the global maxima.

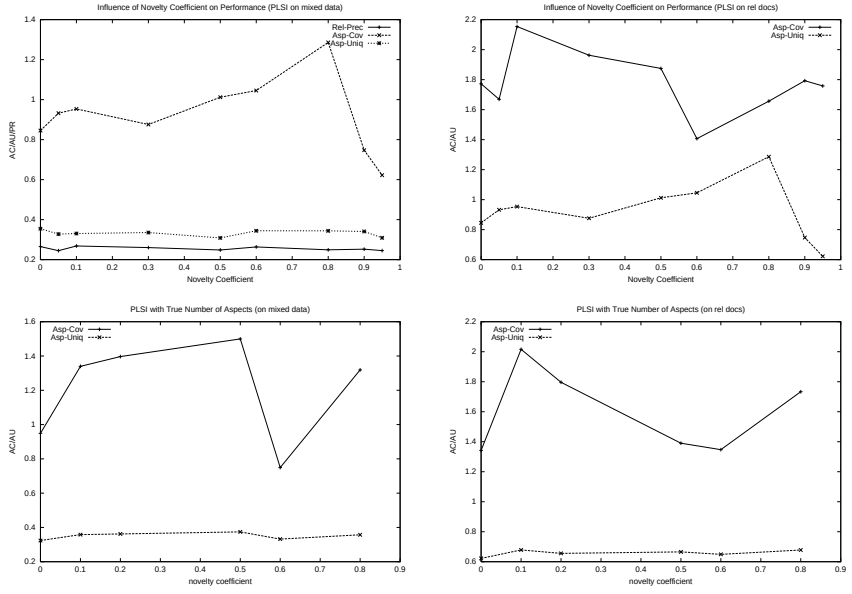

Figure 7.12: Influence of novelty coefficient on performance with the 10 aspects (top) and with the true number of aspects (bottom) on mixed data (left) and relevant data (right).

Data Set	Novelty Coefficient	10 aspects		True aspect count	
		RelPrec(μ)	AspCov(μ)	RelPrec(μ)	AspCov (μ)
Mixed Data	$\mu = 0$	0.265(0)	0.845(0)	0.266(0)	0.949(0)
	$\mu \neq 0$	0.249 (0.8)	1.286(0.8)	0.257 (0.5)	1.500 (0.5)
	improve	-6.0%	+52.2%	-3.4%	+58.1%
Relevant Data	$\mu = 0$	1(0)	1.772(0)	1(0)	1.342(0)
	$\mu \neq 0$	1(0.1)	2.153(0.1)	1(0.1)	2.016 (0.1)
	improve	0%	+21.5%	0%	+50.2%

Data Set	Novelty Coefficient	10 aspects		True aspect count	
		RelPrec(μ)	AspUniq (μ)	RelPrec(μ)	AspUniq(μ)
Mixed Data	$\mu = 0$	0.265(0)	0.355(0)	0.266(0)	0.324(0)
	$\mu \neq 0$	0.263 (0.6)	0.344(0.6)	0.257(0.5)	0.374(0.5)
	improve	-0.8%	-3.1%	-3.4%	+15.4%
Relevant Data	$\mu = 0$	1(0)	0.611(0)	1(0)	0.622(0)
	$\mu \neq 0$	1 (0.9)	0.685(0.9)	1(0.1)	0.678 (0.1)
	improve	0%	+12.1%	0%	+9.0%

Table 7.9: Effectiveness of the aspect loss function. Using a non-zero novelty coefficient is shown to improve the aspect coverage (top) and aspect uniqueness (bottom) even though it decreases relevance-based precision.

In Table 7.9, we compare the model's performance at a zero novelty weight and that at a non-zero novelty weight. Such a comparison provides some idea about how effective our loss function is in capturing the dependency among documents. In Table 7.9, we see that using a non-zero novelty coefficient is able to improve both aspect coverage and aspect uniqueness, even though it decreases precision. Such an improvement in aspect performance when decreasing the precision has not been possible for any of the MMR style loss functions, suggesting that the aspect modeling loss functions are more promising in capturing the aspect coverage. Thus, the KL-divergence loss function is effective in capturing the preference for maximizing the aspect coverage. The same table also compares the performance of using a fixed number of aspects (10) with that of using the true number of aspects (as obtained from the aspect judgments). We see that using the true number of aspects ("cheating") is not always better than using just a fixed number of aspects. For example, when measured with aspect coverage, it performs better than using the fixed 10 aspects on the mixed data, but it is worse on the relevant data set. It is unclear why this is so.

A more complete picture of how the performance is influenced by the novelty weight is given in Figure 7.12, where we see that the performance can be quite sensitive to the

setting of the novelty weight, and the optimal setting depends on both the data set and the performance measure.

Ranking Method	Rel. Precision			Aspect Coverage			Aspect Uniqueness		
	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	0.318	0.718	0.217	1.435	2.753	1.561	0.380	0.861	0.363
Best AC ($\mu = 0.8$)	0.249	0.600	0.157	1.286	2.370	1.477	0.344	0.760	0.325
Improve	-21.7%	-16.4%	-27.6%	-10.4%	-13.9%	-5.4%	-9.5%	-11.7%	-10.5%
Best AU ($\mu = 0$)	0.265	0.637	0.176	0.845	1.911	0.828	0.355	0.804	0.343
Improve	-16.7%	-11.3%	-18.9%	-41.1%	-30.6%	-47.0%	-6.6%	-6.6%	-5.5%

Ranking Method	Aspect Coverage			Aspect Uniqueness		
	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	1.962	3.150	2.139	0.647	0.972	0.650
Best AC ($\mu = 0.1$)	2.154	3.430	2.468	0.669	0.953	0.707
Improve	+9.8%	+8.9%	+15.4%	+3.4%	-2.0%	+8.8%
Best AU ($\mu = 0.5$)	1.875	2.987	2.049	0.676	0.975	0.673
Improve	-4.4%	-5.2%	-4.2%	+4.5%	+0.3%	+3.5%

Table 7.10: Effectiveness of Basic PLSI (10 aspects) on mixed data (top) and relevant data (bottom).

In Table 7.10, we compare the best performance of PLSI using 10 aspects with the relevance ranking baselines. We see that although PLSI improves both aspect coverage and aspect uniqueness on the relevant data, it is significantly worse than the baseline on the mixed data, indicating that the capture of relevance is weak using 10 aspects in this basic aspect model. However, the PLSI model appears to have great potential for further improving the performance. For example, when we use a larger number of aspects, we can achieve better aspect uniqueness. This can be seen from Table 7.11, where we show that it is possible to achieve the same aspect uniqueness as the baseline if we use 60 aspects with a non-zero novelty weight ($\mu = 0.1$). Another possibility of improving the basic model is to fix one aspect to a background model and force a constant weight on this special

Ranking Method	Rel. Precision			Aspect Coverage			Aspect Uniqueness		
	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	0.318	0.718	0.217	1.435	2.753	1.561	0.380	0.861	0.363
Best AC (BM)	0.274	0.675	0.174	1.456	2.75	1.588	0.347	0.766	0.351
Improve	-13.8%	-6.0%	-19.8%	+1.5%	-0.1%	+1.7%	-8.7%	-11.0%	-3.3%
Best AU (60 asp)	0.278	0.648	0.177	1.363	2.560	1.540	0.380	0.807	0.372
Improve	-12.6%	-9.7%	-18.4%	-5.0%	-7.0%	-1.3%	0%	-6.3%	+2.5%

Table 7.11: Effectiveness of Improved PLSI. Fixing one aspect to be the background model achieves the best AC when $\mu = 0.1$ and the background weight is set to 0.2.; using 60 aspects achieves the best AU when $\mu = 0.1$.

background aspect. The idea is to force all the other aspects to focus on more specific content aspect, rather than being distracted by the common words. Indeed, as also shown in Table 7.11, when we set the background weight to 0.2 and set $\mu = 0.1$ we can actually improve the aspect coverage over the baseline. Note that the relevance-based precision is actually significantly decreased. Since low precision means that there are relatively fewer relevant documents ranked on the top, the improvement in aspect coverage means that those few relevant documents are probably good relevant documents that can cover many aspects. Thus, unlike the MMR models, which could only improve the aspect coverage when the relevance ranking is accurate, the PLSI model is able to directly maximize the aspect coverage without relying on a good relevance ranking.

Latent Dirichlet Allocation

All the experiments on LDA are based on an implementation using the variational Bayesian approach (Minka and Lafferty, 2002), and the task on the mixed data is to re-rank the top 30 documents returned by a baseline relevance ranking.

We first look at how the novelty coefficient affects the performance when using 10 aspects. This is shown in Figure 7.13. We see that, on both the mixed data and the relevant data, using a non-zero novelty weight can perform better than using a zero weight in terms of the aspect performance. Indeed, as we use a larger novelty weight, the aspect uniqueness tends to improve. However, on the mixed data, we also see that the relevance-based precision decreases as the novelty weight increases, which means that avoiding redundancy may allow some non-relevant documents to be ranked high, thus decreasing the precision. We also see that on the mixed data set, if the novelty weight is too large, the aspect performance decreases significantly, which is also expected.

The fact that a non-zero novelty weight can perform better than a zero weight suggests that the novelty parameter in the loss function is useful for improving aspect coverage. In Table 7.12, we compare the performance of using a zero novelty weight and the best of using a non-zero weight. We see that using a non-zero novelty weight improves aspect coverage and aspect uniqueness for both the mixed data and the relevant data. At the same time, the precision on the mixed data is decreased. Here again, we see that it is possible to improve the aspect performance without improving the relevance precision, which we have not seen with any of the MMR models.

Next, we examine the influence of the number of aspects used in the model. The results

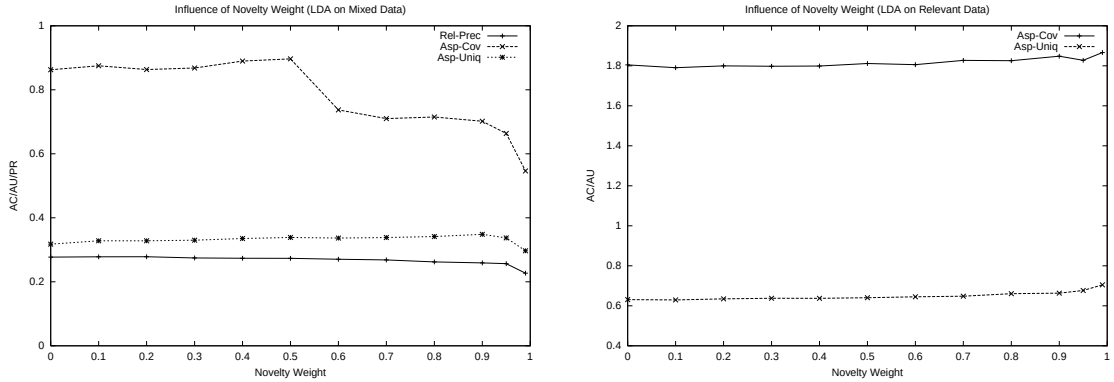

Figure 7.13: Influence of novelty coefficient on performance using LDA with 10 aspects on mixed data (left) and relevant data (right).

Data Set	Novelty Coefficient	Aspect Coverage		Aspect Uniqueness	
		RelPrec(μ)	AspCov(μ)	RelPrec(μ)	AspUniq (μ)
Mixed Data	$\mu = 0$	0.277(0)	0.863(0)	0.277(0)	0.318(0)
	$\mu \neq 0$	0.273 (0.5)	0.897(0.5)	0.259 (0.9)	0.348 (0.9)
	improve	-1.4%	+3.9%	-6.5%	+9.4%
Relevant Data	$\mu = 0$	1(0)	1.804(0)	1(0)	0.631(0)
	$\mu \neq 0$	1(0.99)	1.866(0.99)	1(0.99)	0.705(0.99)
	improve	0%	+3.4%	0%	+11.7%

Table 7.12: Effectiveness of the aspect loss function for LDA. Using a non-zero novelty coefficient is shown to improve the aspect coverage and uniqueness for both the mixed data and the relevant data. On the mixed data, relevance precision is decreased.

are shown in Figure 7.14. It seems that the precision decreases as we increase the number of aspects. The optimal number of aspects for the aspect uniqueness measure is probably around 20, and a larger number of aspects seems to hurt the aspect uniqueness. The optimal number of aspects for the aspect coverage appears to be around 40.

Among the limited experiments that we have performed, the best results from LDA are disappointingly much worse than the baseline results, as shown in Table 7.13, but it does improve the aspect uniqueness on the relevant data set. Our exploration of LDA is rather preliminary. For example, in order to speed up the experiments, we limited to the maximum number of EM iterations to 50, which may have affected the accuracy of the estimated model parameters. Also, the computation of the document and query models are not exact. It is even possible that there are bugs that are not easy to detect in our program. Further exploration of LDA is clearly needed to fully develop the potential of this general

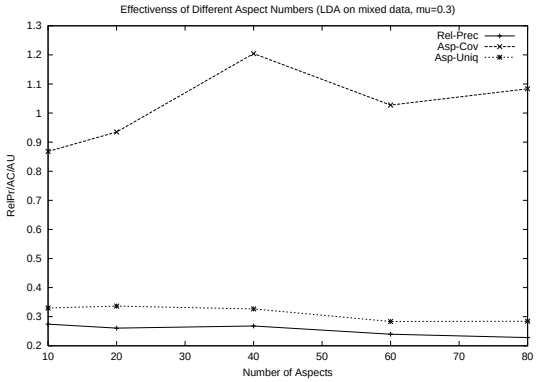

Figure 7.14: Influence of the number of aspects on performance using LDA with $\mu = 0.3$ on mixed data.

aspect language model.

Ranking	Rel. Precision			Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	0.320	0.707	0.214	1.723	3.223	1.991	0.406	0.877	0.423
Best AC ($\mu = 0.3$,40)	0.268	0.627	0.168	1.204	2.152	1.322	0.327	0.657	0.323
Improve	-16.3%	-11.3%	-21.5%	-30.1%	-33.2%	-33.6%	-19.5%	-25.1%	-23.6%
Best AU ($\mu = 0.3$,10)	0.277	0.667	0.185	0.955	1.870	0.957	0.358	0.705	0.365
Improve	-13.4%	-5.7%	-13.6%	-44.6%	-42.0%	-51.9%	-11.8%	-19.6%	-13.7%

Ranking	Aspect Coverage			Aspect Uniqueness		
Method	Avg.	At0.1	At0.5	Avg.	At0.1	At0.5
FBBase	2.268	3.708	2.598	0.661	0.995	0.690
Best AC & AU ($\mu = 0.99$,10)	1.866	2.609	2.130	0.704	0.997	0.719
Improve	-17.7%	+29.6%	-18.0%	+6.5%	+0.2%	+4.2%

Table 7.13: Effectiveness of LDA on mixed data (top) and relevant data (bottom). The number in the parenthesis is the number of aspects used.

7.5 Conclusions and Further Work

In this chapter, we have demonstrated how the risk minimization framework can go beyond the traditional notion of independent topical relevance through the study of a non-traditional retrieval task, i.e., the aspect retrieval problem, within the risk minimization framework. The goal of aspect retrieval is not to retrieve as many relevant documents as possible, but to retrieve as many distinct relevant aspects as possible. For this non-traditional retrieval task, we consider two measures – aspect coverage and aspect uniqueness.

The aspect retrieval problem requires non-traditional ranking methods. We derived two types of aspect retrieval models, both involving a dependent loss function. The first type of models are to increase aspect coverage indirectly by reducing the redundancy among documents, and are essentially Maximal Marginal Relevance (MMR) models based on language models. The second type of models, called Maximal Diverse Relevance (MDR) models, are to increase aspect coverage more directly by modeling the hidden aspects in documents.

We proposed two ways to exploit language modeling to measure redundancy and novelty; one is based on KL-divergence and the other is based on mixture models. Evaluation shows that the mixture model measure is more effective in capturing redundancy, and can achieve significantly better aspect uniqueness than pure relevance ranking when re-ranking relevant documents. The basic idea of this mixture model measure is to represent the known information by a known information language model, and to assume that a new document is generated by a mixture model involving the known information model and a background component model. Our novelty measure (or equivalently redundancy measure) is then the estimated mixing weight on the known information model (or the background model). This novelty measure can be useful for re-ranking any subset of documents to reduce redundancy. Given the success of this simple mixture model, it is worth exploring other more sophisticated mixture models as presented in (Zhang et al., 2002).

For the MMR type of models, we proposed four different ways to combine relevance with novelty. According to the experimental results, no single method is consistently better than the others. Instead, we observed that some of these models are better on the mixed data set while others are better on the relevant data set. A direct combination of the mixture model novelty measure with the regular KL-divergence relevance score is shown to be effective in improving both aspect coverage and aspect uniqueness on the relevant data set, suggesting that it is indeed helpful for improving aspect coverage through redundancy elimination. However, this same method does not perform well on the mixed data set, and in particular, can never perform better than the baseline relevance ranking, suggesting that the redundancy elimination helps improve aspect coverage only when the relevance ranking is accurate. Interestingly, the marginal document model approach, where we directly estimate a language model that represents the novel information contained in a document, performs relatively better on the mixed data set, improving over the relevance baseline by every measure, including relevance-based precision. However, this method does not perform well on the relevant data set. Thus, the improvement on the mixed data set is likely due to the improvement on relevance ranking. Further study is needed to understand why such an aspect-oriented model would actually improve the relevance precision. Overall, the MMR

type of model does not appear to be very effective in improving aspect performance on the mixed data. This may be because the amount of redundancy among relevant documents is relatively low compared with the amount of non-relevant documents in our data set. Since these models are designed to exploit redundancy removal, their effectiveness can be best demonstrated through a highly redundant data set. Further experiments with synthetic data would allow us to test their effectiveness more fairly. Another interesting research direction would be to model redundancy, novelty, and relevance at the passage-level.

For the MDR models, we derived a general aspect retrieval function based on aspect KL-divergence, and studied two variants of the aspect generative models – the Probabilistic Latent Semantic Indexing (PLSI) and the Latent Dirichlet Allocation (LDA) approach. Evaluation shows that, for both PLSI and LDA, our loss function is effective in capturing the dependency among documents, and the optimal performance is obtained through the use of a *non-zero* novelty weight, which means that the goal of *collectively* covering the aspects with multiple documents is effectively reflected in our loss function. One interesting observation is that both PLSI and LDA have shown the capacity of improving the aspect performance when the relevance precision decreases, which has not been observed in our experiments with the MMR models. This means that while these aspect models lose some relevant documents on the top, they could selectively keep on top those good relevant documents that better cover more aspects. We also see that, using a KL-divergence function on the aspect space, our relevance precision with these aspect models tends to be worse than that of applying the KL-divergence function to the word space, and as a result, the absolute performance is generally not as strong as the baseline relevance ranking. Nevertheless, we have shown that it is possible for the PLSI model to perform better than the relevance ranking baseline if we further improve the basic model, e.g., by incorporating a background aspect. This is very encouraging, especially because the aspect performance is increased even when the relevance precision is decreased, which indicates the model's effectiveness in modeling the aspects.

Our study of aspect retrieval is preliminary, but the results have clearly demonstrated the generality of the risk minimization framework. We have shown how we can derive non-traditional ranking methods by using loss functions that go beyond the traditional notion of relevance. Just as in the exploration of models for the standard ad hoc retrieval task, the framework serves as a map that allows us to explore a variety of non-traditional models systematically.

As a study of the aspect retrieval problem, however, our results are far from conclusive.

The improvement over the baseline ranking using only relevance is not always significant, especially in the case of ranking a mixed set of relevant and non-relevant documents. An overall observation on this data set is that the relevance ranking appears to have a quite strong correlation with the aspect coverage measure, which also makes sense intuitively, since the more relevant aspects a document covers, the more likely that it would get a higher score according to any reasonable relevance measure. Indeed, some follow up experiments done by William Cohen on the same data set show that there is really not that much room to improve the aspect retrieval performance over the baseline methods (Cohen, 2002). Thus, an important research direction would be to further examine the behavior of these non-traditional ranking methods using well-controlled synthetic data. This would allow us to observe how the performance is affected by factors such as the average number of aspects in a document and the level of redundancy in the relevant documents. This should also help us better understand the problem of aspect retrieval and the appropriateness of the proposed evaluation measures. Relevance feedback in aspect retrieval is another very interesting direction worth exploring. For example, if we know some information about what aspects might be interesting to the user, such information can be exploited to regulate the aspect model. In terms of specific aspect retrieval models, we have only touched on a very basic formulation of the models, especially in the case of LDA, where a lot of approximation and simplification have been assumed in our experiments. Also, our implementation of PLSI appears to suffer from the problem of multiple local maximas; the performance is highly sensitive to the local maxima. The LDA model is relatively stable, but has not performed as well as the PLSI model. Further study of these models is clearly needed to thoroughly understand the methods and their effectiveness in performing the aspect retrieval task. Finally, it would be interesting to study other alternative loss functions for the aspect retrieval problem; this can be expected to result in completely new aspect retrieval models.

Chapter 8

Conclusions

In this chapter, we summarize the key research results presented in this thesis, and discuss the directions and opportunities for further research.

8.1 Summary

This thesis presents a new general probabilistic framework for text retrieval based on Bayesian decision theory. In this framework, queries and documents are modeled using statistical language models, user preferences are modeled through loss functions, and retrieval is cast as a risk minimization problem. This risk minimization framework not only unifies several existing retrieval models within one general probabilistic framework, but also facilitates the development of new principled approaches to text retrieval through the use of statistical language models. We have explored several interesting special cases of the framework, demonstrating various advantages of this new framework over the traditional retrieval methods.

One fundamental difference between the risk minimization framework and any existing retrieval framework is that the risk minimization framework treats the entire retrieval problem as a *decision* problem, and incorporates *statistical language models* as major components in the framework. While previous work has also treated retrieval from a decision-theoretic view, no previous work has given a *complete* decision-theoretic formal model for retrieval. The risk minimization framework is thus the first complete formal treatment of retrieval in statistical decision theory. This is also the first time a user variable ($\mathcal{U}$) and a document source variable ($\mathcal{S}$) have been explicitly and formally introduced in a retrieval

model. The decision space in the risk minimization framework, in principle, may consist of all the possible actions that the system can take in response to a query, which allows us to treat the retrieval problem in the most general way. Such a general decision-theoretic view explicitly suggests that retrieval can be modeled as an *interactive* process that involves cycles of a user's reformulating the query and the system's presenting information. Indeed, with the risk minimization framework, we can condition the current retrieval decision on all the information about the retrieval context, the user, and the interaction history, to perform context-sensitive retrieval. In contrast, the traditional retrieval models are quite restricted due to their reliance on unrealistic simplification assumptions about relevance (e.g., the independent relevance assumption). They are generally inadequate for handling user factors such as redundancy tolerance and readability, and cannot model an interactive retrieval process without relying on heuristics.

The risk minimization framework makes it possible to systematically and formally study general optimal retrieval strategies. For example, through making different assumptions about the loss function for ranking we have derived an optimal ranking principle, which addresses several limitations of the similar probability ranking principle. Specifically, when assuming an independent loss function and a sequential browsing model, we can show that the optimal ranking is to rank documents according to the expected risk of each document, which can be computed *independently* for each document. An interesting implication is that such a ranking is optimal whether the user has a high-precision or high-recall retrieval preference.

The general incorporation of statistical language models in a retrieval framework is another important contribution of this thesis. A major challenge in text retrieval has long been to develop principled retrieval approaches that also perform well empirically. Theoretically well-motivated models have rarely led directly to good performance; heuristic retrieval parameters always need to be introduced to increase the flexibility of a model. Moreover, because the parameters are empirically motivated, heavy empirical tuning of parameters is always necessary to achieve good retrieval performance. In contrast, the retrieval parameters in the risk minimization framework are generally introduced as part of a statistical language model. This makes it possible to exploit statistical estimation methods to improve retrieval performance and set retrieval parameters automatically. As a special case of the risk minimization framework, we presented a two-stage language modeling approach that can achieve excellent retrieval performance through setting retrieval parameters completely automatically.

With the risk minimization framework, this thesis extends the existing work on the language modeling approach to information retrieval in several ways. First, we studied the smoothing of language models. We show a general connection between the language modeling approach and several heuristics used in traditional models, including the TF-IDF weighting and document length normalization. We extensively evaluated several popular smoothing methods (Jelinek-Mercer, Dirichlet priors, and absolute discounting), and found that the retrieval performance is generally sensitive to the smoothing parameters and that smoothing plays two different roles, with one role being to improve the accuracy of the estimated document language model (the estimation role), while the other to accommodate generation of non-informative common words in the query (the query modeling role). This provides an empirical justification for our two-stage smoothing method, which generalizes the basic language model proposed in (Ponte and Croft, 1998). Second, we studied how to perform feedback in the language modeling approach. Feedback has so far been dealt with heuristically in the language modeling approach with serious conceptual inconsistency. As another instance of the risk minimization framework, we derive a KL-divergence retrieval model, which covers the basic language modeling approach as a special case. The KL-divergence model explicitly deals with the query model and can perform feedback more naturally by treating it as query model updating. We proposed two specific query model updating algorithms based on feedback documents. Evaluation indicates that both algorithms are effective for feedback, and that the updated query models outperform the original query models significantly. This demonstrates the possibility of improving retrieval performance through using more reasonable language models and better estimation methods. This is in contrast with the lack of guidance on how to improve a retrieval model in the previous work.

Due to its generality in formalizing retrieval tasks, the risk minimization retrieval framework further allows for incorporating user factors beyond the traditional notion of topical relevance. Traditionally, it has been hard to formally model such factors as redundancy and sub-topics *within* a retrieval model, though a general linear combination of relevance measure and novelty measure has been given in (Carbonell and Goldstein, 1998). We presented language models that can formally capture redundancy and sub-topics in documents, and study loss functions that can rank documents in terms of both relevance and sub-topic diversity. Evaluation shows that the proposed language models can effectively capture redundancy and can outperform the relevance-based ranking method for the aspect retrieval task. While the study of the aspect retrieval is preliminary, it has clearly demonstrated how the risk minimization framework can be exploited to model such a non-traditional retrieval

problem.

8.2 Future Research Directions

The risk minimization framework opens up many new possibilities for developing principled approaches to text retrieval, and serves as a general framework for applying statistical language models to text retrieval. The special cases explored in this thesis represent only a small step in exploring the full potential of the risk minimization framework. There are many interesting future research directions, including:

Automatic parameter setting: It is always a major scientific challenge to set retrieval parameters automatically without relying on heavy experimentation. One major advantage of using language models is the possibility of estimating retrieval parameters completely automatically. The two-stage smoothing method, along with its parameter estimation methods, is a promising small step toward this goal. However, the two-stage smoothing method is a relatively simple model, which does not take advantage of feedback. It would be very interesting to study how to estimate parameters in more powerful models such as those based on feedback documents.

Document structure analysis: A common assumption in text retrieval is to treat the document as the information unit. However, the relevant information often only represents a small part of a long document. Thus, it is desirable to go further than just retrieving a relevant document, and to *locate* the real relevant information in a relevant document. Traditionally, heuristic approaches have been applied to retrieve passages, which can be regarded as a step toward this goal. But the boundary of relevant information may vary according to different queries, thus it would be interesting to explore a dynamic approach where passage segmentation is integrated with the retrieval function. With the risk minimization framework, it is possible to incorporate segmentation into the loss function and exploit language models such as Hidden Markov Models to model the document structure.

Aspect retrieval models: The aspect retrieval problem provides an interesting setting for studying redundancy elimination and sub-topic modeling in text retrieval, and is often also a more accurate formulation of the retrieval problem when high recall is preferred. In this thesis, we have made some preliminary exploration of language models for the aspect retrieval problem, but there are still many open issues to further explore. In particular, the Latent Dirichlet Allocation model has great potential for modeling the hidden aspects directly, and is certainly worth further exploring. Another interesting research direction is

to develop aspect retrieval models that can support aspect-based feedback.

Interactive retrieval models: In a real retrieval situation, the goal of satisfying a user's information need is often accomplished through a series of interactions between the user and the retrieval system. Such an interactive retrieval process suggests that, at any moment, the retrieval decision may depend on the previous actions that the system and the user have taken. For example, the documents that a user has already seen should be considered in order to avoid redundancy. Indeed, in general, all the information about the previous actions should be utilized to provide a context for making the current retrieval decision. With the risk minimization framework, we can formally incorporate all these variables and derive personalized and context-sensitive interactive retrieval models. It would be very interesting to extend the risk minimization framework to formalize an interactive retrieval process so as to optimize the *global* and *long term* utility over a sequence of retrieval interactions.

Bibliography

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

Berger, J. (1985). *Statistical decision theory and Bayesian analysis*. Springer-Verlap.

Blei, D., Ng, A., and Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Bookstein, A. and Swanson, D. (1975). A decision theoretic foundation for indexing. *Journal for the American Society for Information Science*, 26:45–50.

Buckley, C. (1995). Automatic query expansion using SMART: Trec-3. In Harman, D., editor, *Overview of the Third Text Retrieval Conference (TREC-3)*, pages 69–80. NIST Special Publication 500-225.

Callan, J. P., Croft, W., and Harding, S. (1992). The inquery retrieval system. In *Proceedings of the Third International Conference on Database and Expert System Applications*, pages 78–82. Springer-Verlag.

Carbonell, J. and Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR'98*, pages 335–336.

Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Cohen, W. (2002). Personal communications.

Cooper, W. (1991). Some inconsistencies and misnomers in probabilistic IR. In *Proceedings of SIGIR'91*, pages 57–61.

Cooper, W. S. (1994). The formalism of probability theory in IR: A foundation for an encumbrance? In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 242–247.

Cooper, W. S. and Maron, M. E. (1978). Foundations of probabilistic and utility-theoretic indexing. *Journal of the ACM*, 25(1):67–80.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley.

Croft, W. B. (1981). Document representation in probabilistic models of information retrieval. *Journal of the American Society for Information Science*, pages 451–457.

Croft, W. B. and Harper, D. (1979). Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35:285–295.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statist. Soc. B*, 39:1–38.

Evans, D. A. and Lefferts, R. G. (1994). Design and evaluation of the CLARIT TREC-2 system. In Harman, D., editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 137–150.

Evans, D. A. and Zhai, C. (1996). Noun-phrase analysis in unrestricted text for information retrieval. In *Proceedings of ACL 1996*, pages 17–24.

Fox, E. (1983). *Expending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types*. PhD thesis, Cornell University.

Froehlich, T. J. (1994). Relevance reconsidered - towards an agenda for the 21st century: Introduction to special topic issue on relevance research. *Journal of the American Society for Information Science*, 45(3):124–134.

Fuhr, N. (1992). Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255.

Fuhr, N. (1993). Representations, models and abstractions in probabilistic information retrieval. In Opitz, O., Lausen, B., and Klar, R., editors, *Information and Classification. Concepts, Methods and Applications, Springer, Berlin et al*, pages 259–267.

Fuhr, N. (2001). Language models and uncertain inference in information retrieval. In *Proceedings of the Language Modeling and IR workshop*, pages 6–11.

Fuhr, N. and Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3):223–248.

Fung, R. and Favero, B. D. (1995). Applying Bayesian networks to information retrieval. *Communications of the ACM*, 38(3):42–48.

Gey, F. (1994). Inferring probability of relevance using the method of logistic regression. In *Proceedings of ACM SIGIR'94*, pages 222–231.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40, parts 3,4:237–264.

Harper, D. J. and van Rijsbergen, C. J. (1978). An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation*, 34(3):189–216.

Harter, S. P. (1975). A probabilistic approach to automatic keyword indexing (part I & II). *Journal of the American Society for Information Science*, 26:197–206 (Part I), 280–289 (Part II).

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

Hiemstra, D. and Kraaij, W. (1998). Twenty-one at TREC-7: Ad-hoc and cross-language track. In *Proceedings of Seventh Text REtrieval Conference (TREC-7)*, pages 227–238.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of ACM SIGIR'99*, pages 50–57.

Jelinek, F. (1997). *Statistical methods for speech recognition*. MIT Press.

Kalt, T. (1996). A new probabilistic model of text classification and retrieval. Technical Report 78, CIIR, Univ. of Massachusetts.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35:400–401.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184.

Kwok, K. and Chan, M. (98). Improving two-stage ad-hoc retrieval for short queries. In *Proceedings of SIGIR'98*, pages 250–256.

Kwok, K. L. (1995). A network approach to probabilistic information retrieval. *ACM Transactions on Office Information System*, 13:324–353.

Lafferty, J. and Zhai, C. (2001a). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'01*, pages 111–119.

Lafferty, J. and Zhai, C. (2001b). Probabilistic IR models based on query and document generation. In *Proceedings of the Language Modeling and IR workshop*, pages 1–5.

Lafferty, J. and Zhai, C. (2003). Probabilistic relevance models based on document and query generation. In Croft, W. B. and Lafferty, J., editors, *Language Modeling and Information Retrieval*. Kluwer Academic Publishers.

Lavrenko, V. and Croft, B. (2001). Relevance-based language models. In *Proceedings of SIGIR'01*, pages 120–127.

Lawrence, S. and Giles, C. L. (1999). Accessibility of information on the web. *Nature*, 400:107–109.

Lewis, D. D. (1992). Representation and learning in information retrieval. Technical Report 91-93, Univ. of Massachusetts.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *European Conference on Machine Learning*, pages 4–15.

Lyman, P. and Varian, H. R. (2000). How much information. Retrieved from http://www.sims.berkeley.edu/how-much-info on Oct. 2002.

MacKay, D. and Peto, L. (1995). A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):289–307.

Maron, M. E. and Kuhns, J. L. (1960). On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7:216–244.

McCallum, A. and Nigam, K. (1998). A comparison of event models for Nave Bayes text classification. In *AAAI-1998 Learning for Text Categorization Workshop*, pages 41–48.

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

Minka, T. and Lafferty, J. (2002). Expectation-propagation for the generative aspect model. In *Proceedings of the UAI 2002*, pages 352–359.

Mittendorf, E. and Schauble, P. (1994). Document and passage retrieval based on hidden Markov models. In *Proceedings of SIGIR'94*, pages 318–327.

Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependencies in stochastic language modeling. *Computer Speech and Language*, 8:1–38.

Ney, H., Essen, U., and Kneser, R. (1995). On the estimation of 'small' probabilities by leaving-one-out. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1202–1212.

Ng, K. (2000). A maximum likelihood ratio information retrieval model. In Voorhees, E. and Harman, D., editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 483–492.

Ponte, J. (1998). *A Language Modeling Approach to Information Retrieval*. PhD thesis, Univ. of Massachusetts at Amherst.

Ponte, J. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR'98*, pages 275–281.

Ribeiro, B. A. N. and Muntz, R. (1996). A belief network model for IR. In *Proceedings of SIGIR'96*, pages 253–260.

Ribeiro-Neto, B., Silva, I., and Muntz, R. (2000). Bayesian network models for information retrieval. In Crestani, F. and Pasi, G., editors, *Soft Computing in Information Retrieval: Techniques and Applications*, pages 259–291. Springer Verlag.

Robertson, S. (2002). Personal communication.

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

Robertson, S. and Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR'94*, pages 232–241.

Robertson, S. and Walker, S. (1997). On relevance weights with little relevance information. In *Proceedings of SIGIR'97*, pages 16–24.

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

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

Robertson, S. E. and Walker, S. (1999). Okapi/keenbow at TREC-8. In Voorhees, E. M. and Harman, D. K., editors, *The Eighth Text REtrieval Conference (TREC 8)*. NIST Special Publication 500-246.

Robertson, S. E., Walker, S., Jones, S., M.Hancock-Beaulieu, M., and Gatford, M. (1995). Okapi at TREC-3. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*, pages 109–126.

Rocchio, J. (1971). Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc.

Rosenfeld, R. (2000). Two decades of statistical language modeling: where do we go from here? In *Proceedings of IEEE*, volume 88.

Rousseau, R. (1990). Extended Boolean retrieval: a heuristic approach. In *Proceedings of SIGIR'90*, pages 495–508.

Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523.

Salton, G. and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 44(4):288–297.

Salton, G., Fox, E., and Wu, H. (1983). Extended boolean information retrieval. *The Communications of the ACM*, 26(1):1022–1036.

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

Salton, G., Wong, A., and Yang, C. S. (1975a). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Salton, G., Yang, C. S., and Yu, C. T. (1975b). A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44.

Saracevic, T. (1970). The concept of "relevance" in information science: a historical review. In Saracevic, T., editor, *Introduction to Information Science*, pages 111–151. R. R. Bowker Company.

Sherman, C. (2001). Google fires new salvo in search engine size wars. *SearchDay*, (157). http://searchenginewatch.com/.

Singhal, A. (2001). Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43.

Singhal, A., Buckley, C., and Mitra, M. (1996). Pivoted document length normalization. In *Proceedings of the 1996 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29.

Song, F. and Croft, B. (1999). A general language model for information retrieval. In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 279–280.

Sparck Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments - part 1 and part 2. *Information Processing and Management*, 36(6):779–808 and 809–840.

Sparck Jones, K. and Willett, P., editors (1997). *Readings in Information Retrieval*. Morgan Kaufmann Publishers.

Strzalkowski, T. (1997). NLP track at TREC-5. In Harman, D., editor, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 97–102.

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

van Rijbergen, C. J. (1977). A theoretical basis for theuse of co-occurrence data in information retrieval. *Journal of Documentation*, pages 106–119.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths.

van Rijsbergen, C. J. (1986). A non-classical logic for information retrieval. *The Computer Journal*, 29(6).

Varian, H. R. (1999). Economics and search (invited talk for SIGIR'99). *SIGIR Forum*, 33(3).

Voorhees, E. and Harman, D., editors (2001). *Proceedings of Text REtrieval Conference (TREC1-9)*. NIST Special Publications. http://trec.nist.gov/pubs.html.

Wong, S. K. M. and Yao, Y. Y. (1989). A probability distribution model for information retrieval. *Information Processing and Management*, 25(1):39–53.

Wong, S. K. M. and Yao, Y. Y. (1995). On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):69–99.

Wurman, R. S. (1989). *Information Anxiety*. Doubleday.

Xu, J. and Croft, W. (1996). Query expansion using local and global document analysis. In *Proceedings of the SIGIR'96*, pages 4–11.

Zhai, C. (1997). Fast statistical parsing of noun phrases for document indexing. In *5th Conference on Applied Natural Language Processing (ANLP-97)*, pages 312–319.

Zhai, C. and Lafferty, J. (2001a). Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410.

Zhai, C. and Lafferty, J. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'01*, pages 334–342.

Zhai, C. and Lafferty, J. (2002). Two-stage language models for information retrieval. In *Proceedings of SIGIR'02*, pages 49–56.

Zhang, Y., Callan, J., and Minka, T. (2002). Redundancy detection in adaptive filtering. In *Proceedings of SIGIR'02*, pages 81–88.