A Study of Smoothing Methods for Language Models Applied to Information Retrieval

CHENGXIANG ZHAI and JOHN LAFFERTY
Carnegie Mellon University

Language modeling approaches to information retrieval are attractive and promising because they connect the problem of retrieval with that of language model estimation, which has been studied extensively in other application areas such as speech recognition. The basic idea of these approaches is to estimate a language model for each document, and to then rank documents by the likelihood of the query according to the estimated language model. A central issue in language model estimation is *smoothing*, the problem of adjusting the maximum likelihood estimator to compensate for data sparseness. In this article, we study the problem of language model smoothing and its influence on retrieval performance. We examine the sensitivity of retrieval performance to the smoothing parameters and compare several popular smoothing methods on different test collections. Experimental results show that not only is the retrieval performance generally sensitive to the smoothing parameters, but also the sensitivity pattern is affected by the query type, with performance being 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 role—to make the estimated document language model more accurate and to "explain" the noninformative words in the query. In order to decouple these two distinct roles of smoothing, we propose a two-stage smoothing strategy, which yields better sensitivity patterns and facilitates the setting of smoothing parameters automatically. We further propose methods for estimating the smoothing parameters automatically. Evaluation on five different databases and four types of queries indicates that the two-stage smoothing method with the proposed parameter estimation methods consistently gives retrieval performance that is close to—or better than—the best results achieved using a single smoothing method and exhaustive parameter search on the test data.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*performance evaluation (efficiency and effectiveness)*

Portions of this work appeared as ZHAI, C. AND LAFFERTY, J. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of ACM SIGIR '01*. ACM, New York, 334–342, and ZHAI, C. AND LAFFERTY, J. 2002. Two-stage language models for information retrieval. In *Proceedings of ACM SIGIR '02*. ACM, New York, 49–56.
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, contract MDA904-00-C-2106.
Authors' addresses: C. Zhai, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801; email: czhai@cs.uiuc.edu; J. Lafferty, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213; email: lafferty@cs.cmu.edu.

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Statistical language models, term weighting, TF-IDF weighting, Dirichlet prior smoothing, Jelinek–Mercer smoothing, absolute discounting smoothing, backoff smoothing, interpolation smoothing, risk minimization, two-stage smoothing, leave-one-out, EM algorithm

1. INTRODUCTION

The study of information retrieval models has a long history. Over the decades, many different types of retrieval models have been proposed and tested. A great diversity of approaches and methodology has been developed, rather than a single unified retrieval model that has proven to be most effective; however, 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 and probabilistic models (e.g., van Rijsbergen [1986], Fuhr [1992], Robertson et al. [1981], and Wong and Yao [1995]). On the other hand, there have been many empirical studies of models, including variants of the vector space model (e.g., Salton and Buckley [1988, 1990] and Singhal et al. [1996]). In some cases, there have been theoretically motivated models that also perform well empirically; for example, the BM25 retrieval function, motivated by the 2-Poisson probabilistic retrieval model, has proven to be quite effective in practice [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; Berger and Lafferty 1999; Miller et al. 1999; Hiemstra and Kraaij 1999]. The basic idea behind the new approach is extremely simple—estimate a language model for each document, and rank documents by the likelihood of the query according to the language model. Yet this new framework 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.

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 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.

In this article, we study the problem of language model smoothing in the context of ad hoc retrieval, focusing on the smoothing of document language models. The research questions that motivate this work are (1) how sensitive is retrieval performance to the smoothing of a document language model? and (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 noninformative words in a query.

In order to decouple these two distinct roles of smoothing, we propose a two-stage smoothing strategy, which can reveal more meaningful sensitivity patterns and facilitate the setting of smoothing parameters. We further propose methods for estimating the smoothing parameters automatically. Evaluation on five different databases and four types of queries indicates that the two-stage smoothing method with the proposed parameter estimation methods consistently gives retrieval performance that is close to—or better than—the best results achieved using a single smoothing method and exhaustive parameter search on the test data.

The rest of the article is organized as follows: In Section 2, we discuss the language modeling approach and its connection with some of the heuristics used in traditional retrieval models. In Section 3, we describe the three smoothing methods we evaluated. The major experiments and results are presented in Sections 4 through 7. In Section 8, we present further experimental results to support the dual role of smoothing and motivate a two-stage smoothing method, which is presented in Section 9. In Section 10 and Section 11, we describe parameter estimation methods for two-stage smoothing, and discuss experimental results on their effectiveness. Section 12 presents conclusions and outlines directions for future work.

2. THE LANGUAGE MODELING APPROACH

In the language modeling approach to information retrieval, one considers the probability of a query as being "generated" by a probabilistic model based on a document. For a query $q = q_1 q_2 \cdots q_n$ and document $d = d_1 d_2 \ldots d_m$, this probability is denoted by $p(q \mid d)$. In order to rank documents, we are interested in estimating the posterior probability $p(d \mid q)$, which from Bayes' formula is given by

$$p(d \mid q) \propto p(q \mid d) p(d), \tag{1}$$

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 1999; Song and Croft 1999]. In other cases, $p(d)$ can be used to capture nontextual information, for example, the length of a document or links in a web page, as well as other format/style features of a document; see Miller et al. [1999] and Kraaij et al. [2002] for empirical studies that exploit simple alternative priors.

In our study, we assume a uniform document prior in order to focus on the effect of smoothing. With a uniform prior, the retrieval model reduces to the calculation of $p(q \mid d)$, which is where language modeling comes in. The language model used in most previous work is the unigram model.[1] This is the multinomial model that assigns the probability

$$p(q \mid d) = \prod_{i=1}^{n} p(q_i \mid d). \tag{2}$$

Clearly, the retrieval problem is now essentially reduced to a unigram language model estimation problem. In this article, we focus on unigram models only; see Miller et al. [1999] and Song and Croft [1999] for 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 helps to gain insight into the language modeling approach.

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

$$\begin{aligned}
\log p(q \mid d) &= \sum_{i=1}^{n} \log p(q_i \mid d) && (3)\\
&= \sum_{i:c(q_i;d)>0} \log p_s(q_i \mid d) + \sum_{i:c(q_i;d)=0} \log p_u(q_i \mid d) && (4)\\
&= \sum_{i:c(q_i;d)>0} \log \frac{p_s(q_i \mid d)}{p_u(q_i \mid d)} + \sum_{i=1}^{n} \log p_u(q_i \mid d) && (5)
\end{aligned}$$

[1]The work of 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, for example, 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

$$\log p(q \mid d) = \sum_{i:c(q_i;d)>0} \log \frac{p_s(q_i \mid d)}{\alpha_d \; p(q_i \mid \mathcal{C})} + n \log \alpha_d + \sum_{i=1}^{n} \log p(q_i \mid \mathcal{C}). \tag{6}$$

Note that the last term on the righthand side is independent of the document 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 between 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 $\frac{p_s(q_i \mid d)}{\alpha_d \; p(q_i \mid \mathcal{C})}$, 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 as a consequence has 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 [1999].

3. SMOOTHING METHODS

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

$$p_{ml}(w \mid d) \;=\; \frac{c(w;d)}{\sum_{w' \in V} c(w';d)}, \tag{7}$$

where V is the set of all words in the vocabulary.

However, the maximum likelihood estimator will generally under estimate the probability of any word unseen in the document, and so the main purpose of smoothing is to assign a nonzero 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, smoothing methods discount the probabilities of the words seen in the text and assign the extra probability mass to the unseen words according to some "fallback" model. For information retrieval, it makes sense 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:

$$p(w \mid d) \;=\; \begin{cases} p_s(w \mid d) & \text{if word } w \text{ is seen} \\ \alpha_d \; p(w \mid \mathcal{C}) & \text{otherwise,} \end{cases} \tag{8}$$

where $p_s(w \mid d)$ 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; if $p_s(w \mid d)$ is given, we must have

$$\alpha_d \;=\; \frac{1 - \sum_{w \in V : c(w;d) > 0} p_s(w \mid d)}{1 - \sum_{w \in V : c(w;d) > 0} p(w \mid \mathcal{C})}. \tag{9}$$

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

A smoothing method may be as simple as adding an extra count to every word, which is called additive or Laplace smoothing, or more sophisticated as in Katz smoothing, where words of different count 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 efficient to implement. Although these three methods 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:

$$p_\lambda(w \mid d) \;=\; (1 - \lambda)\, p_{ml}(w \mid d) + \lambda p(w \mid \mathcal{C}). \tag{10}$$

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 [Jelinek and Mercer 1980]).

Bayesian Smoothing using Dirichlet Priors. A language model is a multinomial distribution, for which the conjugate prior for Bayesian analysis is the Dirichlet distribution [MacKay and Peto 1995]. We choose the parameters of the Dirichlet to be

$$(\mu p(w_1 \mid \mathcal{C}),\; \mu p(w_2 \mid \mathcal{C}), \ldots, \mu p(w_n \mid \mathcal{C})). \tag{11}$$

Table I. Summary of the Three Primary Smoothing Methods Compared in this Article

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	$\frac{c(w;d) + \mu\, p(w \mid \mathcal{C})}{\lvert d \rvert + \mu}$	$\frac{\mu}{\lvert d \rvert + \mu}$	μ
Absolute discount	$\frac{\max(c(w;d) - \delta, 0)}{\lvert d \rvert} + \frac{\delta \lvert d \rvert_u}{\lvert d \rvert} p(w \mid \mathcal{C})$	$\frac{\delta \lvert d \rvert_u}{\lvert d \rvert}$	δ

Thus, the model is given by

$$p_\mu(w \mid d) \;=\; \frac{c(w;d) + \mu p(w \mid \mathcal{C})}{\sum_{w' \in V} c(w';d) + \mu}. \tag{12}$$

The Laplace method is a special case of this technique.

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

$$p_\delta(w \mid d) \;=\; \frac{\max(c(w;d) - \delta, 0)}{\sum_{w' \in V} c(w';d)} + \sigma p(w \mid \mathcal{C}), \tag{13}$$

where $\delta \in [0, 1]$ is a discount constant and $\sigma = \delta \lvert d \rvert_u / \lvert d \rvert$, so that all probabilities sum to one. Here $\lvert d \rvert_u$ is the number of *unique* terms in document d, and $\lvert d \rvert$ is the total count of words in the document, so that $\lvert d \rvert = \sum_{w' \in V} c(w';d)$.

The three methods are summarized in Table I in terms of $p_s(w \mid d)$ 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 above three methods can be implemented very efficiently, assuming that the smoothing parameter is given in advance. The αs can be precomputed 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. Indeed, the scoring complexity for a query q is $O(k \; \lvert q \rvert)$, where $\lvert q \rvert$ 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. EXPERIMENTAL SETUP

Our first goal is to study and compare the behavior of smoothing 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 to compare algorithms. We use five databases from TREC, including three of the largest testing collections for ad hoc retrieval: (1) Financial Times on disk 4, (2) FBIS on disk 5, (3) Los Angeles Times on disk 5, (4) Disk 4 and disk 5 minus Congressional Record, used for the TREC7 and TREC8 ad hoc tasks, and (5) the TREC8 web data.

Table II. Labels used for Test Collections (top), Statistics of the Five Text Collections used in Our Study (middle), and Queries used for Testing (bottom)

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

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

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 interaction of smoothing and query length/type, we use two different versions of each set of queries: (1) title only and (2) a long version (title + description + narrative). 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. Although stop words are often also removed at the preprocessing time, we deliberately index *all* of the words in the collections, 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. Stemming, on the other hand, is unlikely to affect our results on sensitivity analysis; indeed, we could have skipped stemming as well.

In Table II, 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 article, we follow the standard TREC evaluation procedure for ad hoc retrieval, that is, for each topic, the performance figures are computed based on the top 1,000 documents in the retrieval results. For the purpose of studying the behavior of an individual smoothing method, we select a set of representative parameter values and examine the sensitivity of noninterpolated average precision to the variation in these values. For the purpose of comparing smoothing methods, we first optimize the performance of each method using the noninterpolated 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.[2]

5. BEHAVIOR OF INDIVIDUAL METHODS

In this section, we study the behavior of each smoothing method. We first analyze the 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 noninterpolated average precision against different values of the smoothing parameter.

5.1 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 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 terms. This means that documents that match more query terms will 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 $(1-\lambda)p_{ml}(q_i|d)/(\lambda p(q_i\,|\,\mathcal{C}))$, 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 1 show the average precision for different settings of λ, for both large and small collections. It is evident that the precision is 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), 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

[2]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%.

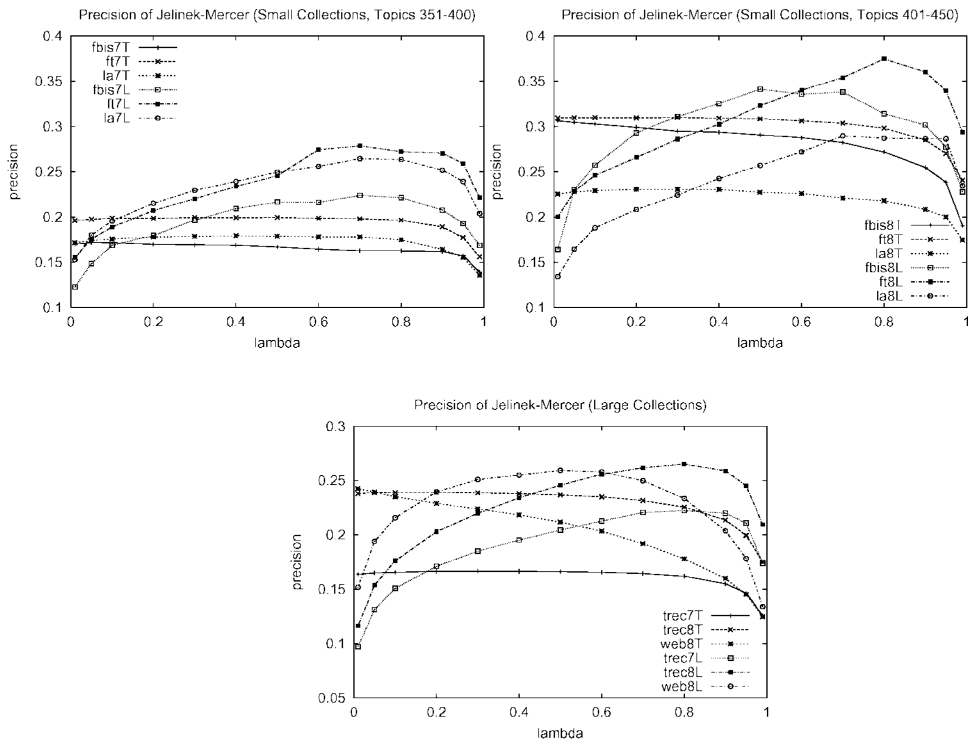

Fig. 1. Performance of Jelinek–Mercer smoothing.

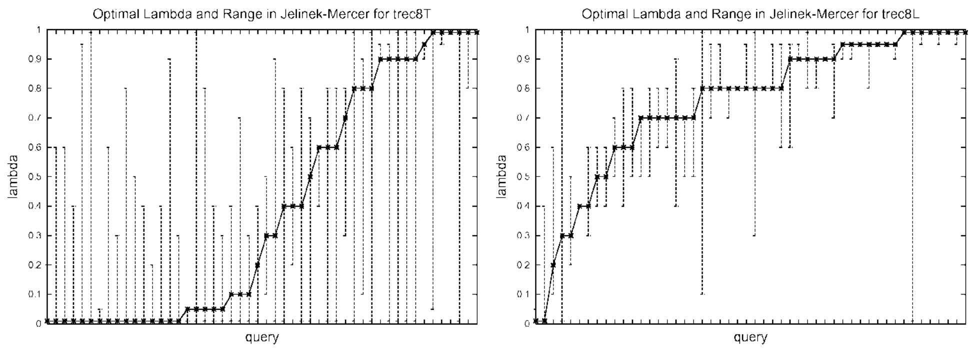

Fig. 2. Optimal λ range for Trec8T (left) and Trec8L (right) in Jelinek–Mercer smoothing. The line shows the optimal value of λ and the bars are the optimal ranges, where the average precision differs by no more than 1% from the optimal value.

the performance achieved by treating the query as a disjunctive query. Thus, the shape of the curves in Figure 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 sensitivity pattern can be seen more clearly from the per-topic plot given in Figure 2, which shows the range of λ values giving models that deviate from the optimal average precision by no more than 1%.

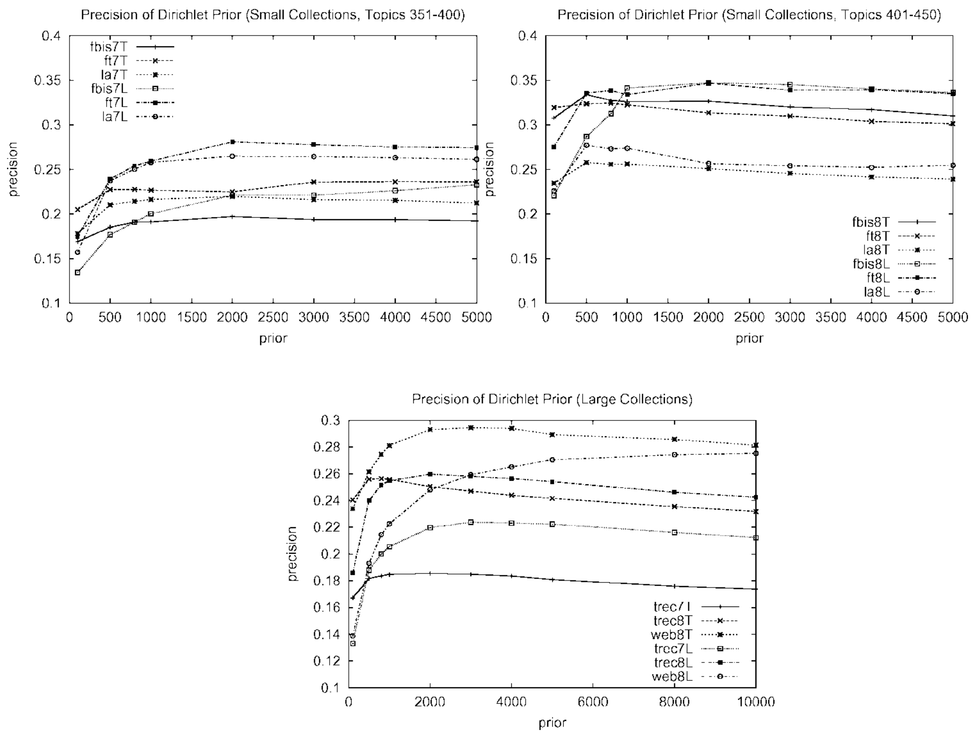

Fig. 3. Performance of Dirichlet smoothing.

5.2 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 $\log(1+c(q_i;d)/(\mu p(q_i\,|\,\mathcal{C})))$. Note that in the Jelinek–Mercer method, the term weight has a document length normalization implicit in $p_s(q_i\,|\,d)$, 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 $\log(1+|d|\,p_{ml}(q_i\,|\,d)/(\mu p(q_i\,|\,\mathcal{C})))$ we see that $|d|/\mu$ is playing the same role as $(1-\lambda)/\lambda$ is playing in Jelinek–Mercer, 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.

The plots in Figure 3 show the average precision for different settings of the prior sample size μ. The precision is again more sensitive to μ for long queries than for title queries, especially when μ is small. Indeed, when μ is sufficiently large, all long queries perform better than title queries, but when μ is very small, it is the opposite. The *optimal* value of μ also tends to be larger for long

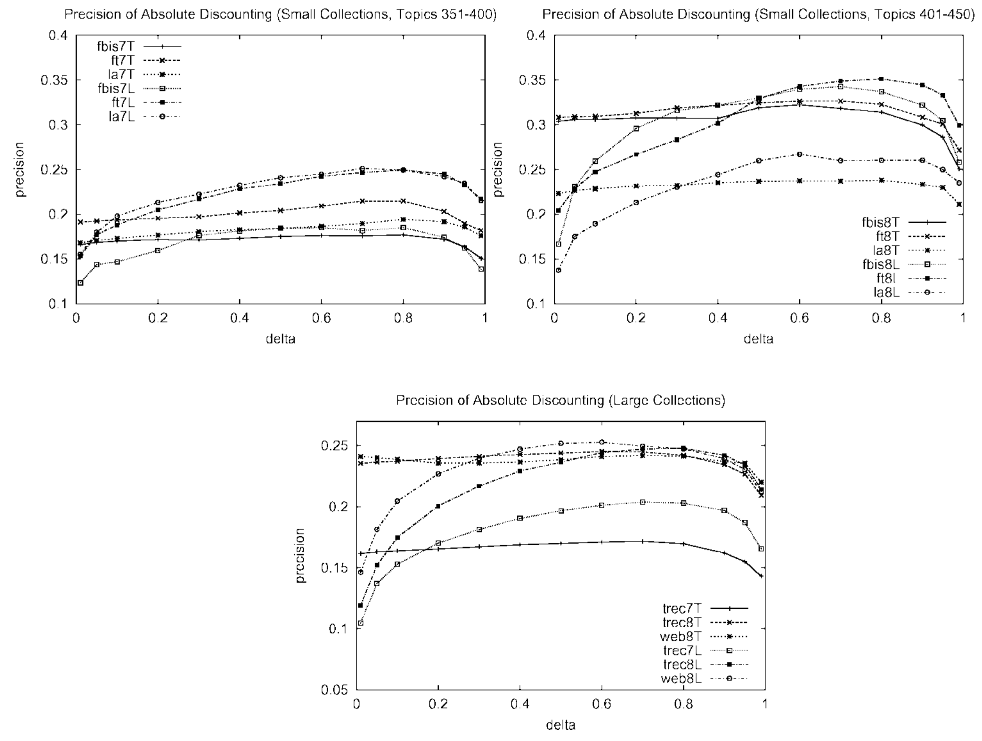

Fig. 4. Performance of absolute discounting.

queries than for title 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.

5.3 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, that is, 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 $\log(1 + (c(q_i; d) - \delta)/(\delta |d|_u p(q_i \mid \mathcal{C})))$. The influence of δ on relative term weighting depends on $|d|_u$ and $p(\cdot \mid \mathcal{C})$, in the following way. If $|d|_u\, p(w \mid \mathcal{C}) > 1$, 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 $p(w \mid \mathcal{C}) < 1/|d|_u$.

The plots in Figure 4 show the average precision for different settings of the discount constant δ. Once again it is clear that the precision is much more sensitive to δ for long queries than for title queries. Similar to Dirichlet prior 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 the 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 perform much better than the long queries on the web collection for Dirichlet prior. Further analysis and evaluation are needed to understand this observation.

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 of this approach is that a high count word may actually end up with more than its actual count in the document, if it is frequent in the fallback model. 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 the 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 $p_{dml}(w)$; (2) a collection language model term, that is, $\alpha_d\ p(w\,|\,\mathcal{C})$. 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 $p_s(w) = p_{dml}(w) + \alpha_d\ p(w\,|\,\mathcal{C})$ and $p_u(w) = \alpha_d\ p(w\,|\,\mathcal{C})$, then the backoff version is given by $p'_s(w) = p_{dml}(w)$ and $p'_u(w) = \frac{\alpha_d\ p(w\,|\,\mathcal{C})}{1-\sum_{w'\in V:c(w';d)>0} p(w'\,|\,\mathcal{C})}$. The form of the ranking formula and the smoothing parameters remain the same. It is easy to see that the parameter α_d in the backoff version differs from that in the interpolation version by a document-dependent term that 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).

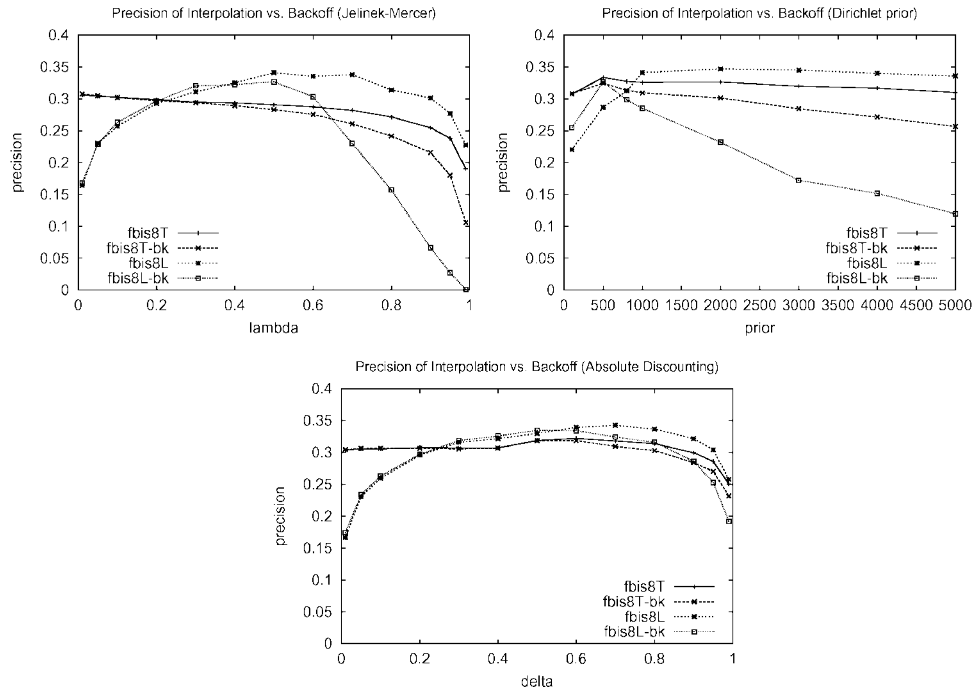

Fig. 5. Interpolation versus backoff for Jelinek–Mercer (top), Dirichlet smoothing (middle), and absolute discounting (bottom).

The results are shown in Figure 5. We find that the backoff performance is more sensitive to the smoothing parameter than that of interpolation, especially in Jelinek–Mercer and Dirichlet prior. The difference is clearly less significant in the absolute discounting method, and this may be due to its lower upper bound ($|d|_u/|d|$) for the original α_d, which restricts the aggressiveness in penalizing long documents. In general, the backoff strategy gives 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.

7. COMPARISON OF METHODS

To compare the three smoothing methods, we select a best run (in terms of noninterpolated 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 III and Table IV 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: Dirichlet prior is better than absolute discounting, which is better than Jelinek–Mercer. In fact, 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

Table III. Comparison of Smoothing Methods on Title Queries

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

The best performance is shown in bold font. 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.

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; many nonoptimal 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 Dirichlet 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 Dirichlet clearly have a better average precision than absolute discounting.

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

Table IV. Comparison of Smoothing Methods on Long Queries

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

The Best Performance is shown in bold font. 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.

Table V. Comparing Long Queries with Short Queries

Method	Query	Avg. Prec.	Prec@10	Prec@20
Jelnek–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%**

Star (*) Indicates the improvement is statistically significant in accordance with Wilcoxin signed rank test at the level of 0.05.

on title queries (except 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 increase of performance is most significant for Jelinek–Mercer, which 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 three smaller databases with that on the large one. We find that the noninterpolated 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 the three small databases. However, the precision at 10 (or 20) documents on large collections is all 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 somehow 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.

8. INFLUENCE OF QUERY LENGTH AND VERBOSITY ON SMOOTHING

In this section, we present experimental results that further clarify which factor is responsible for the high sensitivity observed on long queries. We design experiments to examine two query factors—length and "verbosity." We consider four different types of queries: short keyword, long keyword, short verbose, and long verbose queries. As will be shown, the high sensitivity is indeed caused by the presence of common words in the query.

8.1 Setup

The four types of queries used in our experiments are generated from TREC topics 1-150. These 150 topics are special because, unlike other TREC topics, 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 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
```

Fig. 6. Example topic, number 52. The keywords are used as the “long keyword” version of our queries.

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)[3]
(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).

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.

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

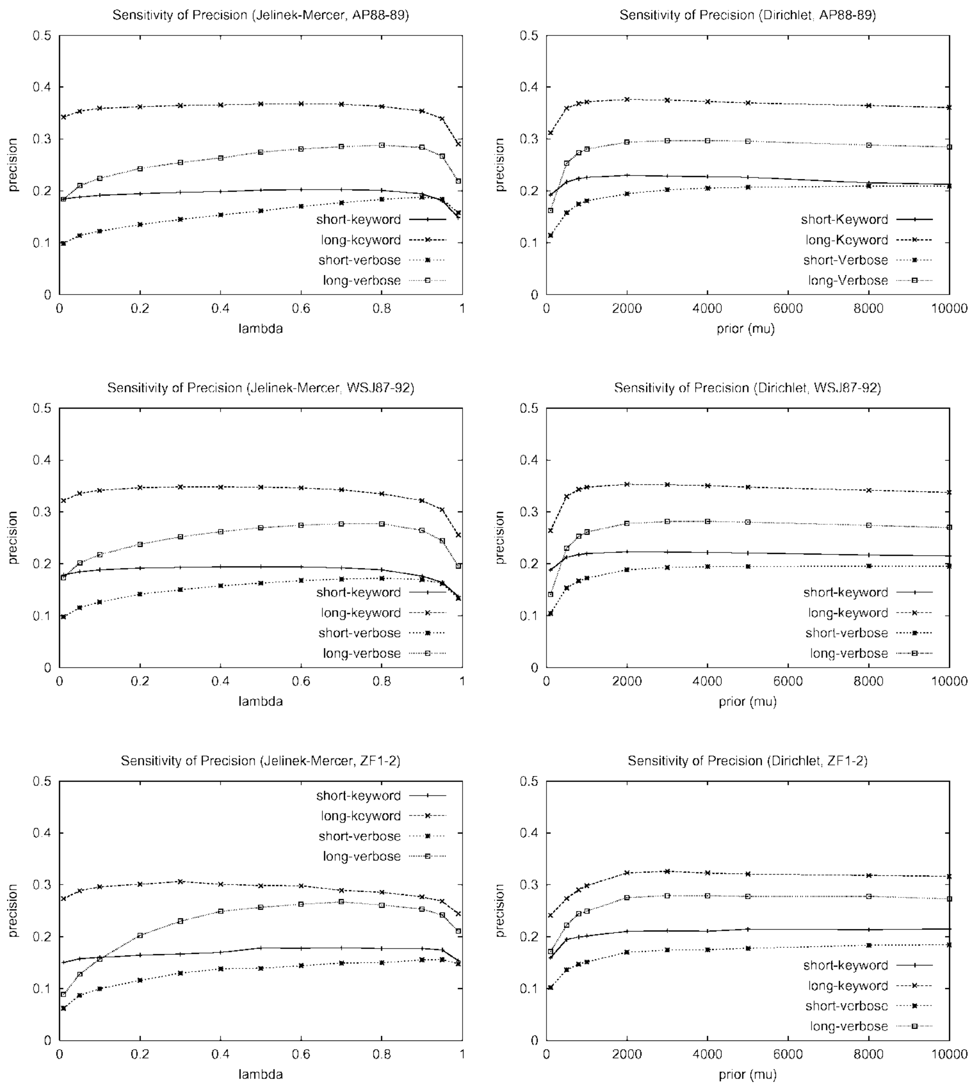

Fig. 7. Sensitivity of Precision for Jelinek–Mercer smoothing (left) and Dirichlet prior smoothing (right) on AP88-89 (top), WSJ87-92 (middle), and ZF1-2 (bottom).

8.2 Results

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 Figure 7. For each query type, we plot 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. The short verbose queries are clearly more sensitive than the long keyword queries. Therefore, the sensitivity is much more correlated with the verbosity of the query than with the length of the query; In all cases, insufficient smoothing is much 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.

9. TWO-STAGE SMOOTHING

The above empirical results suggest that there are two different reasons for smoothing. The first is to address the small sample problem and to explain the unobserved words in a document, and the second is to explain the common/noisy words in a query. In other words, 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 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 mixed effect 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 7 can be interpreted in the following way: The fact that the Dirichlet prior method performs the best on concise title queries suggests that it is good for the estimation role, and the fact that Jelinek–Mercer performs the worst for title queries, but the best for long verbose queries suggests that Jelinek–Mercer is good for the role of query modeling. Intuitively this also makes sense, as 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.

This analysis suggests the following two-stage smoothing strategy. At the first stage, a document language model is smoothed using a Dirichlet prior, and in the second stage it is further smoothed using Jelinek–Mercer. The combined smoothing function is given by

$$p_{\lambda,\mu}(w \mid d) = (1-\lambda)\frac{c(w;d) + \mu p(w \mid \mathcal{C})}{|d| + \mu} + \lambda p(w \mid \mathcal{U}), \tag{14}$$

where $p(\cdot \mid \mathcal{U})$ is the user's query background language model, λ is the Jelinek–Mercer smoothing parameter, and μ is the Dirichlet prior parameter. The combination of these smoothing procedures is aimed at addressing the dual role of smoothing.

The query background model $p(\cdot \mid \mathcal{U})$ is, in general, different from the collection language model $p(\cdot \mid \mathcal{C})$. With insufficient data to estimate $p(\cdot \mid \mathcal{U})$, however, we can assume that $p(\cdot \mid \mathcal{C})$ 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. Clearly when $\lambda = 0$ the model reduces to Dirichlet prior smoothing, whereas when $\mu = 0$, it is Jelinek–Mercer smoothing. Since the combined smoothing formula still follows the general smoothing scheme discussed in Section 3, it can be implemented very efficiently.

9.1 Empirical Justification

In two-stage smoothing we now have two, rather than one, parameter to worry about; however, the two parameters λ and μ are now more meaningful. λ is intended to be a query-related parameter, and should be larger if the query is more verbose. It can be roughly interpreted as modeling the expected "noise" in the query. μ is a document related parameter, and controls the amount of probability mass assigned to unseen words. Given a document collection, the optimal value of μ is expected to be relatively stable. We now present empirical results which demonstrate this to be the case.

Figure 8 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 Trec8 web task).[4] The three lines in each figure correspond to (1) Dirichlet smoothing with title queries; (2) Dirichlet smoothing with long queries; (3) Two-stage smoothing (Dirichlet + Jelinek–Mercer, $\lambda = 0.7$) with long queries.[5] 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.

[4]These are exactly the same as trec7T, trec7L, trec8T, trec8L, web8T, and web8L described in Section 4, but are labelled differently here in order to clarify the effect of two-stage smoothing.

[5]0.7 is a near optimal value of λ when Jelinek–Mercer method is applied alone.

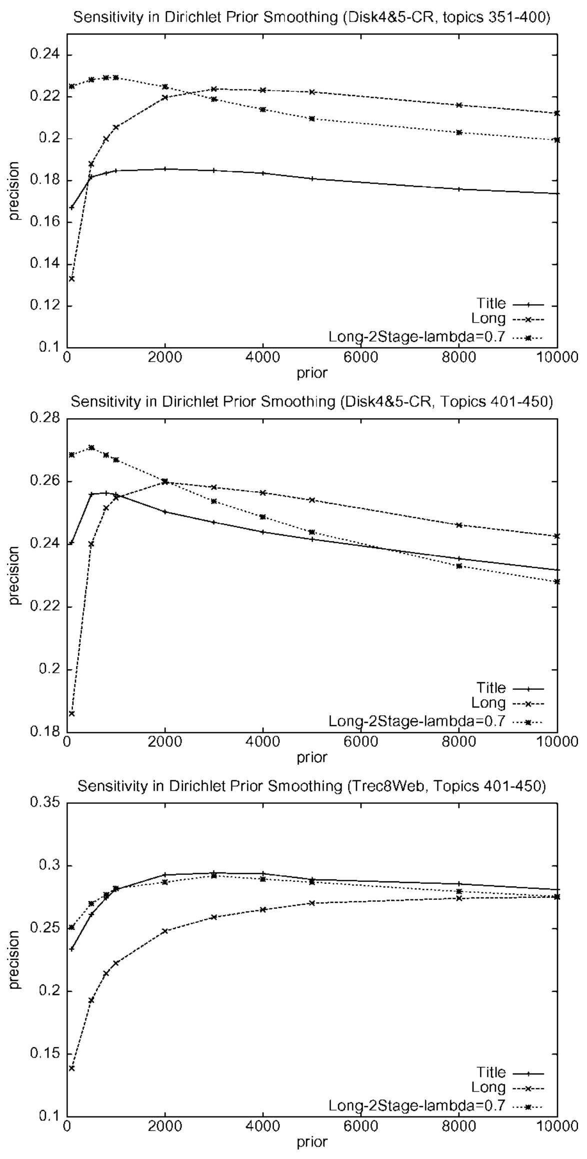

Fig. 8. Two-stage smoothing reveals a more consistent sensitivity pattern than single-stage Dirichlet smoothing for both title queries and long queries. Each figure is for a different collection: Trec7 (top), Trec8 (middle), and web testing collection (bottom).

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 to play the second role of smoothing, the sensitivity pattern of Dirichlet smoothing is much

Table VI. Comparison of the Average Precision between Near Optimal Two-Stage Smoothing and the Best One-Stage Smoothing

Collection	Best JM		Best JM		Two-Stage	
TREC7L	0.222	$\lambda = 0.8$	0.224	$\mu = 3,000$	0.229	$\lambda = 0.7,\ \mu = 800$
TREC8L	0.265	$\lambda = 0.8$	0.260	$\mu = 2,000$	0.268	$\lambda = 0.7,\ \mu = 800$
WEB8L	0.259	$\lambda = 0.5$	0.275	$\mu = 10,000$	0.292	$\lambda = 0.7,\ \mu = 3,000$
TREC7T	0.167	$\lambda = 0.3$	0.186	$\mu = 2,000$	0.184	$\lambda = 0,\ \mu = 800$
TREC8T	0.239	$\lambda = 0.2$	0.256	$\mu = 800$	0.256	$\lambda = 0,\ \mu = 800$
WEB8T	0.243	$\lambda = 0.01$	0.294	$\mu = 3,000$	0.294	$\lambda = 0,\ \mu = 3,000$

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 better than or as well as the *best* single-stage smoothing method.

more similar for title queries and long queries, that is, it becomes much less sensitive to the type/length of queries. By factoring out the second role of smoothing, we are able to see a more consistent and stable behavior of Dirichlet smoothing, which is playing the first role. Indeed, we can compare 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 prior 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 Dirichlet 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 Dirichlet 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 optimal prior is different.

It should also be noted in these figures that with the help from Jelinek–Mercer 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 VI).

It is worth mentioning that on the web data (Figure 8, 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 8, 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 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.

9.2 Theoretical Derivation

We now show that the two-stage smoothing method can also be formally derived using the risk minimization retrieval framework proposed in Lafferty and Zhai [2001]. We first derive a family of general two-stage language models, and then show that our two-stage smoothing method is a special case of using specific generative models for the query and documents. The formal derivation also naturally suggests the parameter estimation methods to be presented in the next section.

9.2.1 *Two-Stage Language Models.* The original language modeling approach as proposed in Ponte and Croft [1998] involves a two-step scoring procedure: (1) Estimate a document language model for each document; (2) Compute the query likelihood using the estimated document language model (directly). The two-stage language modeling approach is a generalization of this two-step procedure, in which a query language model is introduced so that the query likelihood is computed using a query model that is based on the estimated document model, instead of using the estimated document model directly. The use of an explicit and separate query model makes it possible to factor out any influence of queries on the smoothing parameters for document language models.

We now derive the family of two-stage language models for information retrieval using the risk minimization framework, which is a general probabilistic retrieval framework based on Bayesian decision theory. The framework unifies several existing retrieval models within one general probabilistic framework, and facilitates the development of new principled approaches to text retrieval [Lafferty and Zhai 2001].

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. Operationally, documents are ranked based on the following risk function:

$$R(d;q) = \sum_{R\in\{0,1\}} \int_{\Theta_Q} \int_{\Theta_D} L(\theta_Q, \theta_D, R)\, p(\theta_Q \mid q, \mathcal{U})\, p(\theta_D \mid d, \mathcal{S})\, p(R \mid \theta_Q, \theta_D)\, d\theta_D\, d\theta_Q$$

where d denotes a document; q denotes a query; $\mathcal{U}$ is a user variable; $\mathcal{S}$ is a document source variable; θ_Q and θ_D are query language model and document language model respectively; R is a binary relevance variable; and $L(\theta_Q, \theta_D, R)$ is a loss function. The risk associated with retrieving document d in response to query q is seen to be the expected loss given the query, document, and information about the user and document source.

The loss function encodes retrieval preferences and, in general, may depend on all of θ_Q, θ_D, and R. In Lafferty and Zhai [2001], it has been shown that different specific retrieval models can be derived using different choices of loss functions. Here we derive the two-stage language modeling approach using the following loss function, indexed by a small constant ϵ,

$$L_\epsilon(\theta_Q, \theta_D, R) = \begin{cases} 0 & \text{if } \Delta(\theta_Q, \theta_D) \le \epsilon \\ c & \text{otherwise,} \end{cases}$$

where $\Delta : \Theta_Q \times \Theta_D \to \mathbb{R}$ 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. This loss function implicitly depends on the relevance variable R and encodes a user's desire for retrieving documents whose models are close to the query model.

Using this loss function, we obtain the following risk:

$$R(d;q) = c - \int_{\Theta_D} \int_{\theta_Q \in S_\epsilon(\theta_D)} p(\theta_Q \,|\, q, \mathcal{U})\, p(\theta_D \,|\, d, \mathcal{S})\, d\theta_Q\, d\theta_D,$$

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

Now, assuming that $p(\theta_D \,|\, d, \mathcal{S})$ 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 $A \overset{\text{rank}}{\approx} B$ to mean that A and B have the same effect for ranking, we have that

$$R(d;q) \overset{\text{rank}}{\approx} -\int_{\theta_Q \in S_\epsilon(\hat{\theta}_D)} p(\theta_Q \,|\, q, \mathcal{U})\, d\theta_Q$$

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,

$$R(d;q) \overset{\text{rank}}{\approx} -p(\hat{\theta}_D \,|\, q, \mathcal{U}).$$

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

$$p(\hat{\theta}_D \,|\, q, \mathcal{U}) \propto p(q \,|\, \hat{\theta}_D, \mathcal{U})\, p(\hat{\theta}_D \,|\, \mathcal{U}). \tag{15}$$

Equation 15 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: $p(q \,|\, \hat{\theta}_D, \mathcal{U})$ captures how well the estimated document model $\hat{\theta}_D$ explains the query, whereas $p(\hat{\theta}_D \,|\, \mathcal{U})$ 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 paper we assume a uniform prior.

The generic two-stage language model can be refined by specifying a concrete model $p(d \,|\, \theta_D, \mathcal{S})$ for generating documents and a concrete model $p(q \,|\, \theta_Q, \mathcal{U})$ 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. Next, we present the generative models that lead to the two-stage smoothing method described at the beginning of this section.

9.2.2 *The Two-Stage Smoothing Method.* Let $d = d_1 d_2 \cdots d_m$ denote a document, $q = q_1 q_2 \cdots q_n$ denote a query, and $V = \{w_1, \ldots, w_{|V|}\}$ denote the words in the vocabulary. We consider the case where both θ_Q and θ_D are parameters of unigram language models, that is, 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 $p(\cdot \,|\, \theta_D)$, or

$$p(d \,|\, \theta_D, \mathcal{S}) = \prod_{i=1}^{m} p(d_i \,|\, \theta_D).$$

Each document is assumed to be generated from a potentially different model as assumed in the general risk minimization framework. Given a particular document 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

$$\mathrm{Dir}\,(\theta \,|\, \alpha) \;=\; \frac{\Gamma\left(\sum_{i=1}^{|V|} \alpha_i\right)}{\prod_{i=1}^{|V|} \Gamma(\alpha_i)} \prod_{i=1}^{|V|} \theta_i^{\alpha_i - 1}. \tag{16}$$

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

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

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

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

where $|d| = \sum_{w \in V} c(w, d)$ is the length of d. This is the Dirichlet prior smoothing method described in Section 3.

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 Section 8, 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 nondiscriminative words in a query, we assume that a query is 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,

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

where λ is a parameter, roughly indicating the amount of "noise" in q.

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

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

This is precisely the two-stage smoothing formula, in which 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.

10. PARAMETER ESTIMATION

In this section, we present methods for estimating the two parameters μ and λ involved in the two-stage smoothing method.

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 [1998]. Salton and Buckley [1988] 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. 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. 1975] 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 being able to introduce *different* retrieval parameters for queries and documents when appropriate. Indeed, the two-stage smoothing strategy explicitly captures the different influences of the query and document collection on the optimal settings of smoothing parameters, making it possible to set the parameters automatically through statistical estimation methods. Next, 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.

10.1 Estimating μ

The purpose of the Dirichlet prior smoothing at 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 nonzero 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 $\mathcal{C} = \{d_1, d_2, \ldots, d_N\}$ be the collection of documents. Using the Dirichlet smoothing formula, the leave-one-out log-likelihood can be written as

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

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

$$\mu^{(k+1)} = \mu^{(k)} - \frac{g\left(\mu^{(k)}\right)}{g'\left(\mu^{(k)}\right)}$$

Table VII. Estimated Values of μ Along with Database Characteristics

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

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

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

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.

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

10.2 Estimating λ

With the query model hidden, the query likelihood is

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

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

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

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

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

and

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

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 a *single* document. This is because for any given λ, there always is a smoothed document model that assigns the highest likelihood for the query, thus setting $\pi_i = 1$ for this document gives a higher overall likelihood than any other assignment to π_i (V. Lavrenko, 2002, Personal communication). Clearly, this is not desirable and empirically it leads to non-optimal performance. To address this problem, we initialize $\{\pi_i\}_{i=1}^N$ with a *uniform* distribution and λ with 0.5, and allow use early stopping in the EM algorithm, allowing a maximum of 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. The performance is not very sensitive to the exact number of EM iterations, as long as it is not too large (e.g., smaller than 20); it also appears to be "safer" to use a smaller number of iterations, in the sense that the performance will not be too much different from the optimal performance.

11. 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 8.2). 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.[6] 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 preprocessing—only a Porter stemmer was used, and no stop words were 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 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 VIII and Table IX 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

[6]We label these collections differently here to make the labelling consistent with that in Table VIII.

Table VIII. Comparison of the Estimated Two-Stage Smoothing with the Best Single Stage Smoothing Methods on Small Collections

Collection	Query	Method	Avg. Prec.	(Median)	InitPr	Pr@10	Pr@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**	**0.524**
		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**

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.

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 single smoothing methods, we also show (in parentheses) the

Table IX. Comparison of the Estimated Two-Stage Smoothing with the Best Single Stage Smoothing Methods on Large Collections

Collection	Query	Method	Avg. Prec.	(Median)	InitPr	Pr@10	Pr@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**

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.

median average precision for all the parameter values that are tried.[7] We see that for Jelinek–Mercer the sensitivity is clearly higher on verbose queries than on keyword queries; the median is usually much 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 Figure 7, 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 VIII and Table IX, that such a simple approach would not perform so well as our parameter estimation methods. Indeed, the two-stage

[7]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}.

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 2001; 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 there is no ad hoc parameter tuning involved in the retrieval process with the approach. 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.

12. CONCLUSIONS

We have studied the problem of language model smoothing in the context of information retrieval. By rewriting the query-likelihood retrieval model 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 (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 find that the retrieval performance is generally sensitive to the smoothing parameters, suggesting that an understanding and appropriate setting of smoothing parameters is very important in the language modeling approach.

We have made several interesting observations that help us understand each of the methods better. The Jelinek–Mercer method generally performs well, but tends to perform much better for verbose queries than for keyword queries. The optimal value of λ has a strong correlation with the query type. For keyword queries, the optimal value is generally very small (around 0.1), while for 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 keyword queries than for 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 verbose queries. The absolute discounting method performs well on keyword queries, but not very well on 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 version.

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 noninformative common words in the query (query modeling role). This dual role of smoothing can also explain why the backoff versions of all the smoothing methods do not work well; it is because the query modeling role is not well supported in these methods.

The experimental results further suggest that 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 Dirichlet prior with Jelinek–Mercer and explicitly decouples the two roles of smoothing (Dirichlet prior for the estimation role and Jelinek–Mercer for the second). We provide empirical evidence to show that the two-stage smoothing method indeed results in a more meaningful pattern of the smoothing parameters. Specifically, with two-stage smoothing, the optimal setting of Jelinek–Mercer λ is more correlated with the query while that of Dirichlet prior μ is more related to the document collection. This suggests that it is possible to optimize μ based on only the document collection without depending on queries and similarly optimize λ primarily based on queries.

In order to estimate the smoothing parameters automatically in the two-stage smoothing method, we formally study this method in the risk minimization framework. We show that such a two-stage smoothing method is a special case of a family of more general two-stage language models, which can be derived from the risk minimization retrieval framework by using a special loss function. The derivation suggests that we can use 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 single smoothing method, achievable only through an exhaustive parameter search.

The effectiveness and robustness of the two-stage smoothing approach, along with the fact that there is no ad hoc parameter tuning involved, make it a solid baseline approach for evaluating retrieval models.

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], and Kneser–Ney smoothing [Kneser and Ney 1995]). Second, while we have shown that the automatic two-stage smoothing gives retrieval performance close to the best results attainable using a single smoothing method, we have not yet analyzed the optimality of the estimated parameter values in the two-stage parameter space. It would be interesting to see the relative optimality of the estimated μ and λ when fixing one of them. Measures such as perplexity can also be used to gauge the optimality of the estimated parameters. Third, 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 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 subtopics that a user has already encountered (e.g., through reading previously retrieved results), in order to focus ranking on the *new* subtopics in a relevant set of documents.

ACKNOWLEDGMENTS

We thank Victor Lavrenko, Rong Jin, Jamie Callan, W. Bruce Croft, and the anonymous reviewers for helpful comments on this work.

REFERENCES

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*. ACM, New York, 222–229.

Chen, S. F. and Goodman, J. 1998. An empirical study of smoothing techniques for language modeling. Tech. Rep. TR-10-98, Harvard University.

Fuhr, N. 1992. Probabilistic models in information retrieval. *Comput J. 35*, 3, 243–255.

Good, I. J. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika 40, parts 3, 4*, 237–264.

Hiemstra, D. and Kraaij, W. 1999. Twenty-one at TREC-7: Ad-hoc and cross-language track. In *Proceedings of 7th Text REtrieval Conference (TREC-7)*. 227–238.

Jelinek, F. and Mercer, R. 1980. Interpolated estimation of markov sourceparameters from sparse data. In *Pattern Recognition in Practice*, E. S. Gelsema and L. N. Kanal, Eds. 381–402.

Katz, S. M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. 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*. IEEE Computer Society Press, Los Alamitos, Calif., 181–184.

Kraaij, W., Westerveld, T., and Hiemstra, D. 2002. The importance of prior probabilities for entry page search. In *Proceedings of SIGIR'02*. ACM, New York, 27–34.

KWOK, K. AND CHAN, M. 1998. Improving two-stage ad-hoc retrieval for short queries. In *Proceedings of SIGIR'98*. ACM, New York, 250–256.

LAFFERTY, J. AND ZHAI, C. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'01*. ACM, New York, 111–119.

LAVRENKO, V. AND CROFT, B. 2001. Relevance-based language models. In *Proceedings of SIGIR'01*. ACM, New York, 120–127.

MACKAY, D. AND PETO, L. 1995. A hierarchical Dirichlet language model. *Nat. Lang. Eng. 1*, 3, 289–307.

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*. ACM, New York, 214–221.

NEY, H., ESSEN, U., AND KNESER, R. 1994. On structuring probabilistic dependencies in stochastic language modeling. *Comput. Speech Lang. 8*, 1–38.

NEY, H., ESSEN, U., AND KNESER, R. 1995. On the estimation of 'small' probabilities by leaving-one-out. *IEEE Trans. Pattern Anal. Machine Intel. 17*, 12, 1202–1212.

PONTE, J. 1998. A language modeling approach to information retrieval. Ph.D. dissertation. 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*. ACM, New York, 275–281.

ROBERTSON, S. E., VAN RIJSBERGEN, C. J., AND PORTER, M. F. 1981. Probabilistic models of indexing and searching. In *Information Retrieval Research*, R. N. Oddy et al., Eds. Butterworths, 35–56.

ROBERTSON, S. E., WALKER, S., JONES, S., M.HANCOCK-BEAULIEU, M., AND GATFORD, M. 1995. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, D. K. Harman, Ed. 109–126.

SALTON, G. AND BUCKLEY, C. 1988. Term-weighting approaches in automatic text retrieval. *Inf. Proc. Manage. 24*, 513–523.

SALTON, G. AND BUCKLEY, C. 1990. Improving retrieval performance by relevance feedback. *J. Amer. Soc. Inf. Sci. 44*, 4, 288–297.

SALTON, G., WONG, A., AND YANG, C. S. 1975. A vector space model for automatic indexing. *Commun. ACM 18*, 11, 613–620.

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*. ACM, New York, 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*. ACM, New York, 279–280.

SPARCK JONES, K. AND WILLETT, P., Eds. 1997. *Readings in Information Retrieval*. Morgan-Kaufmann, San Mateo, Calif.

VAN RIJSBERGEN, C. J. 1986. A non-classical logic for information retrieval. *Comput. J. 29*, 6, 481–485.

VOORHEES, E. AND HARMAN, D., Eds. 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. 1995. On modeling information retrieval with probabilistic inference. *ACM Trans. Inf. Syst. 13*, 1, 69–99.

ZHAI, C. AND LAFFERTY, J. 2001. Model-based feedback in the KL-divergence retrieval model. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001)*. 403–410.

Received Auguest 2002; revised June 2003; accepted August 2003