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

Chengxiang Zhai
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

John Lafferty
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

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 then rank documents by the likelihood of the query according to the estimated language model. A core problem in language model estimation is *smoothing*, which adjusts the maximum likelihood estimator so as to correct the inaccuracy due to data sparseness. In this paper, 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.

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 [21]. There has been a great diversity of approaches and methodology 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., [14, 3, 15, 22]). On the other hand, there have been many empirical studies of models, including many variants of the vector space model (e.g., [17, 18, 19]). 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 [16].

Recently, a new approach based on language modeling has been successfully applied to the problem of ad hoc retrieval [13, 1, 10, 5]. 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 a zero probability to unseen words. When estimating a language model based on a limited amount of text, such as a single document, smoothing of the maximum likelihood model is extremely important. Indeed, many language modeling techniques are centered around the issue of smoothing. In the language modeling approach to retrieval, the accuracy of smoothing is directly related to the 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 the retrieval performance is affected by the choice of a smoothing method and its parameters.

In this paper, 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? (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 effect of smoothing is very sensitive to the type of queries, which 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 generation of common and non-informative words in a query. Different smoothing methods have behaved somewhat dif-

SIGIR'01, September 9-12, 2001, New Orleans, Louisiana, USA

ferently. Some methods tend to perform better for concise title queries while others tend to perform better for long verbose queries. Moreover, some methods are more stable than others, in the sense that their performance is less sensitive to the choice of parameters.

2. THE LANGUAGE MODELING APPROACH

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

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

As discussed in [1], the righthand side of the above equation has an interesting interpretation, where, $p(d)$ is our prior belief that d is relevant to any query and $p(q \mid d)$ is the query likelihood given the document, which captures how well the document "fits" the particular query q.

In the simplest case, $p(d)$ is assumed to be uniform, and so does not affect document ranking. This assumption has been taken in most existing work [1, 13, 12, 5, 20]. In other cases, $p(d)$ can be used to capture non-textual information, e.g., the length of a document or links in a web page, as well as other format/style features of a document. In our study, we assume a uniform $p(d)$ in order to focus on the effect of smoothing. See [10] for an empirical study that exploits simple alternative priors.

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

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

Clearly, the retrieval problem is now essentially reduced to a unigram language model estimation problem. In this paper we focus on unigram models only; see [10, 20] for some explorations of bigram and trigram models.

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

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

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

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

where n is the length of the query. Note that the last term on the righthand side is independent of the document 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.

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

The connection just derived shows that the use of the collection language model as a reference model for smoothing document language models implies a retrieval formula that implements TF-IDF weighting heuristics and document length normalization. This suggests that smoothing plays a key role in the language modeling approaches to retrieval. A more restrictive derivation of the connection was given in [5].

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 simplest method is the maximum likelihood estimate, simply given by relative counts

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

[1] The work of Ponte and Croft [13] adopts something similar to, but slightly different from the standard unigram model.

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

Table 1: Summary of the three primary smoothing methods compared in this paper.

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 non-zero probability to the unseen words and improve the accuracy of word probability estimation in general.

There are many smoothing methods that have been proposed, mostly in the context of speech recognition tasks [2]. In general, all smoothing methods are trying to discount the probabilities of the words seen in the text, and to then assign the extra probability mass to the unseen words according to some "fallback" model. For information retrieval, it makes much sense, and is very common, to exploit the collection language model as the fallback model. Following [2], we assume the general form of a smoothed model to be the following:

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

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

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

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

A smoothing method may be as simple as adding an extra count to every word (called additive, or Laplace smoothing), or more sophisticated as in Katz smoothing, where words 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 relatively efficient to implement. We excluded some well-known methods, such as Katz smoothing [7] and Good-Turing estimation [4], because of the efficiency constraint[2]. Although the methods we evaluated are simple, the issues that they bring to light are relevant to more advanced methods. The three methods are described below.

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

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

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

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

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

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

Thus, the model is given by

$$p_\mu(w \mid d) \;=\; \frac{c(w;d) + \mu\, p(w \mid \mathcal{C})}{\sum_w c(w;d) + \mu} \qquad (2)$$

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 [11]. 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

$$p_\delta(w \mid d) \;=\; \frac{\max(c(w;d) - \delta, 0)}{\sum_w c(w;d)} + \sigma p(w \mid \mathcal{C}) \qquad (3)$$

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 $\lvert d \rvert = \sum_w c(w;d)$.

The three methods are summarized in Table 1 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 three methods can be implemented very efficiently, when the smoothing parameter is given in advance. The α's can be pre-computed for all documents at index time. The weight of a matched term w can be computed easily based on the collection language model $p(w \mid \mathcal{C})$, the query term frequency $c(w;q)$, the document term frequency $c(w;d)$, and the smoothing parameters. 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 goal is to study the behavior of individual smoothing methods as well as to compare different methods. As is well-known, the performance of a retrieval algorithm may vary significantly according to the testing collection used. It is generally desirable to have larger collections and more queries. We use the following five databases from TREC, including three of the largest testing collections for ad hoc retrieval, i.e., the official TREC7 ad hoc, TREC8 ad hoc, and

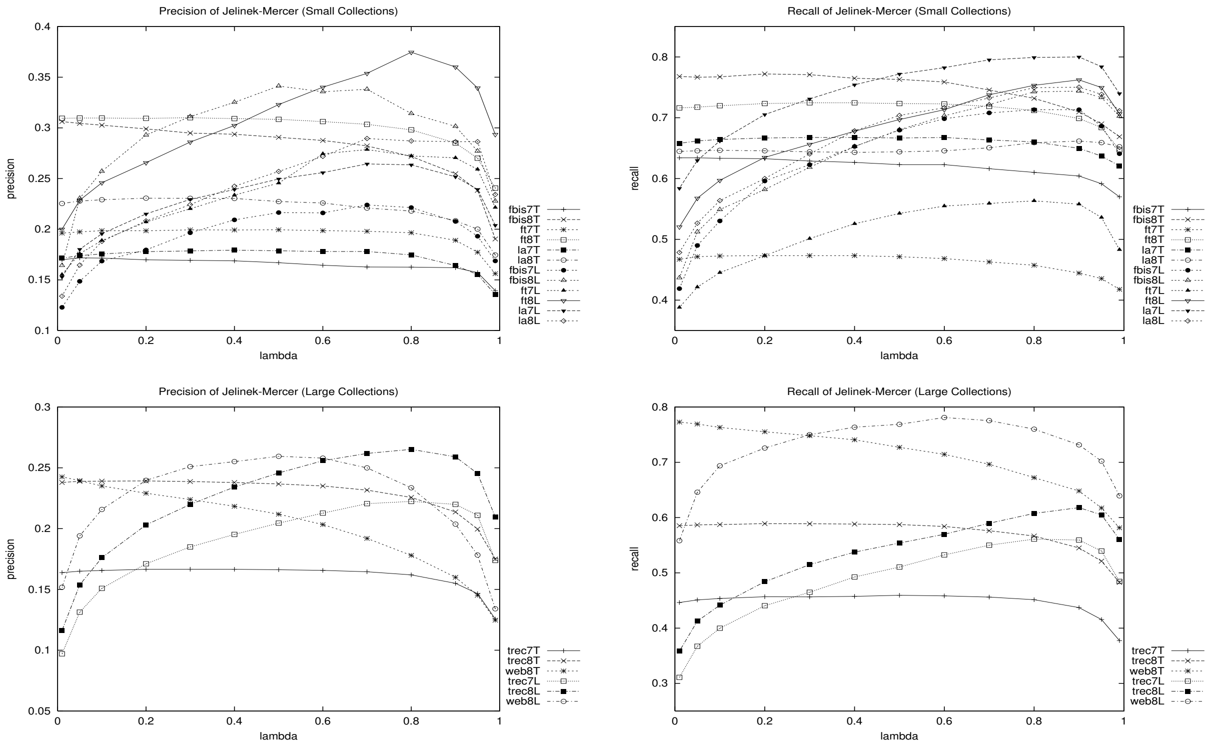

Figure 1: Performance of Jelinek-Mercer smoothing.

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

Table 2: Labels used for test collections.

TREC8 web track testing collections: (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 CR, used for the TREC7 and TREC8 ad hoc tasks, and (5) the TREC8 Web data.

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, (2) 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. We deliberately indexed all the words in the language, since we do not want to be biased by any artificial choice of stop words and we believe that the effects of stop word removal should be better achieved by exploiting language modeling techniques.

In Table 2 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. (While it is certainly possible to set the parameters differently for individual queries and documents through some kind of training procedure, this is out of the scope of the present paper.) 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 precision and recall to the variation in these values. For the purpose of comparing smoothing methods, we first optimize the performance of each method using the non-interpolated average precision as the optimization criterion, and then compare the best runs from each method. The optimal parameter is determined by searching over the entire parameter space.[3]

5. BEHAVIOR OF INDIVIDUAL METHODS

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

Jelinek-Mercer smoothing. When using the Jelinek-Mercer smoothing method with a fixed λ, we see that the parameter α_d in our ranking function (see Section 2) is the same for

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

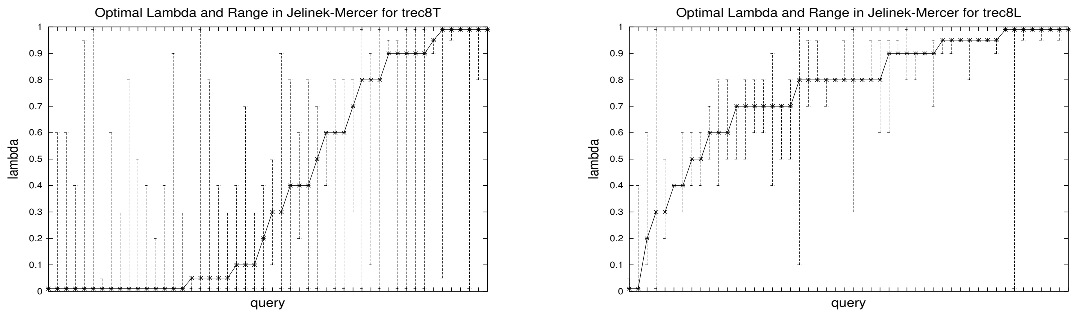

Figure 2: Optimal λ range for trec8t (left) and trec8l (right) in Jelinek-Mercer smoothing. The line shows the optimal value of λ and the bars are the optimal ranges.

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 $\log(1 + (1 - \lambda)p_{ml}(q_i|d)/(\lambda p(q_i \mid \mathcal{C})))$. Thus, a small λ means more emphasis on relative term weighting. Indeed, if λ approaches one, then all term weights tend to zero, and the scoring formula approaches coordination level matching, which is simply the count of matched terms.

The plots in Figure 1 show the average precision and recall for different settings of λ, for both large and small collections. It is evident that both precision and recall are much more sensitive to λ for long queries than for title queries. The web collection, however, is an exception, where performance is very sensitive to smoothing even for title queries. For title queries, the retrieval performance tends to be optimized when λ is small (around 0.1), whereas for long queries, the optimal point is generally higher, and 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 right end of the curve (when λ is very close to one) should be close to the performance of coordination level matching.

Such sensitivity and trend can be seen more clearly from the per-topic plot of the optimal range of λ shown in Figure 2. The optimal range is defined as the maximum range of λ values that deviate from the optimal average precision by no more than 0.01.

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

The plots in Figure 3 show the average precision and recall for different settings of the prior sample size μ. It is again clear that both precision and recall are much more sensitive to μ for long queries than for title queries, especially when μ is small. However, the *optimal* value of μ does not seem to be much different for title queries and long queries. While it still tends to be slightly larger for long queries, 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, as it tends to the performance of coordination level matching.

Absolute discounting. The term weighting behavior of the absolute discounting method is a little more complicated. Obviously, here α_d is also document sensitive. It is larger for a document with a flatter distribution of words, i.e., when the count of unique terms is relatively large. Thus, it penalizes documents with a word distribution highly concentrated on a small number of words. The weight of a matched term is $\log(1+(c(q_i; d)-\delta)/(\delta|d|_u p(q_i \mid \mathcal{C})))$. 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 and recall for different settings of the discount constant δ. Once again it is clear that both precision and recall are much more sensitive to δ for long queries than for title queries. Similar to Bayesian smoothing, but different from Jelinek-Mercer smoothing, the optimal value of δ does not seem to be much different for title queries and long queries. Indeed, the optimal value of δ tends to be around 0.7. This is true not only for both title queries and long queries, but also across all testing collections.

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

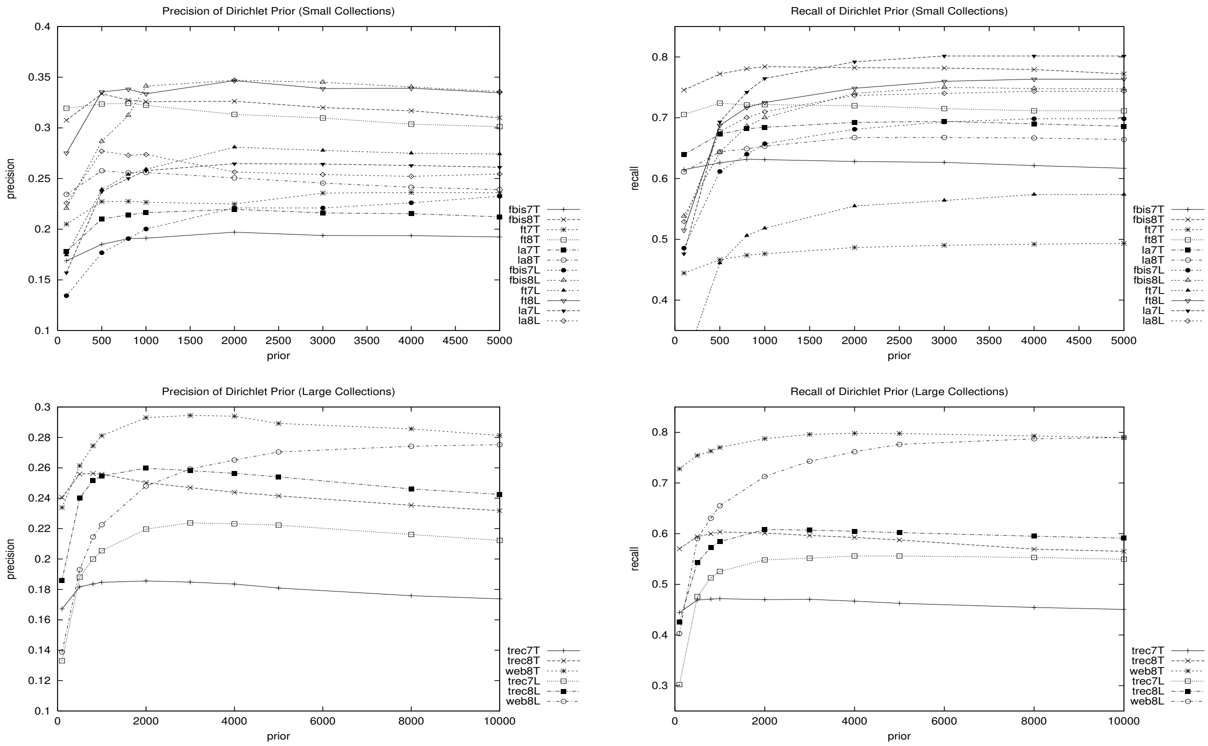

Figure 3: Performance of Dirichlet smoothing.

6. COMPARISON OF METHODS

To compare the three smoothing methods, we select a best run (in terms of non-interpolated average precision) for each 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 3 for both titles and long queries.

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. Indeed, Dirichlet prior has the best average precision in all cases but one. In particular, it performed extremely well on the Web collection, significantly better than the other two. The good performance is relatively insensitive to the choice of μ. Indeed, many non-optimal Dirichlet runs are also significantly better than the optimal runs for Jelinek-Mercer and absolute discounting.

For long queries, there is also a partial order. On average, Jelinek-Mercer is better than Dirichlet and absolute discounting by all three precision measures, though its average precision is almost identical to that of Dirichlet. 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, we see that the three methods all perform better on long queries than on title queries (except that Dirichlet prior performs worse on long queries than title queries on the web collection), but the increase of performance is most significant for Jelinek-Mercer. Indeed, Jelinek-Mercer is the worst for title queries, but the best for long queries. It appears that Jelinek-Mercer is much more effective when queries are 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 non-interpolated average precision on the large database is generally much worse than that on the smaller ones, and is often similar to the worst one among all the three small databases. However, the precision at 10 (or 20) documents on large collections is all significantly better than that on small collections. This is not surprising, since a given precision at a cutoff point of 10 documents would correspond to a much lower level of recall for a large collection than for a small collection. This is exactly the same reason as why one can expect a higher precision at 10 documents when a query has many more relevant documents. For both title queries and long queries, the *relative* performance of each method tends to remain the same when we merge the databases. Interestingly, that 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. One possible explanation is that smoothing actually plays two different roles in the query likelihood retrieval method. One role is to improve the accuracy of the estimated documents language model, and can be referred to as the *estimation* role. The

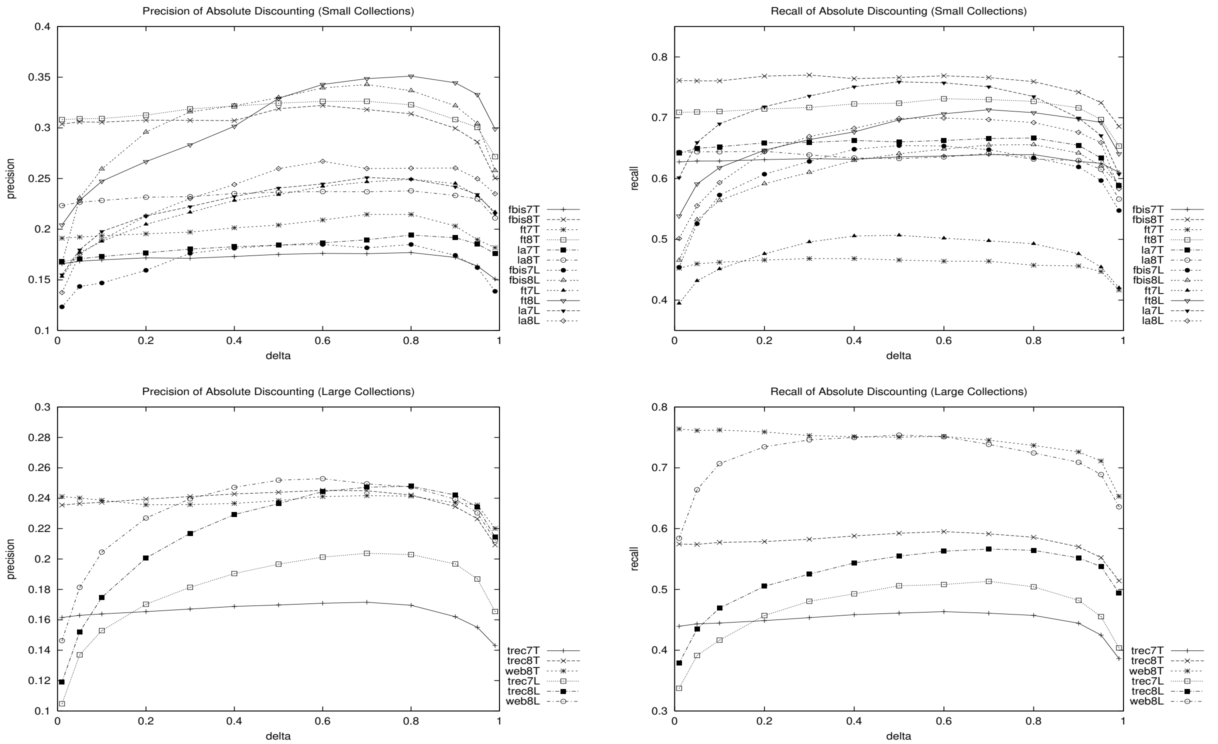

Figure 4: Performance of absolute discounting.

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 [10]. 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 effect of smoothing that we observed is generally a mixed effect of both roles of smoothing. But, for title queries, the effect is more dominated by the estimation role, since in our experiments the title queries have few or no non-informative common words, whereas for long queries, the effect is more influenced by the role of query modeling for they often have many non-informative common words. Thus, the fact that the Dirichlet prior method performs the best on 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 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.

7. 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 [7]. The backoff strategy is very popular in speech recognition tasks.

Following [2], 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, i.e., $\alpha_d\, p(w\,|\,\mathcal{C})$. 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 $p_u'(w) = \frac{\alpha_d\, p(w\,|\,\mathcal{C})}{1-\sum_{i:c(w_i;d)>0} p(w_i\,|\,\mathcal{C})}$. The form of the ranking formula and the smoothing parameters remain the same. It is easy to see that the α_d in the backoff version differs from that in the interpolation version by a document-dependent term which further penalizes long documents. The weight of a matched term due to backoff smoothing has a much wider range of values ($(-\infty, +\infty)$) than that for interpolation ($(0, +\infty)$). Thus, analytically,

Collection	Jelinek-Mercer avgpr, pr@10d, pr@20d (λ)	Dirichlet Prior avgpr, pr@10d, pr@20d (μ)	Absolute Discounting avgpr, pr@10d, pr@20d (δ)
fbis7T	0.172, **0.284**, 0.220 (0.05)	**0.197**, 0.282, **0.238** (2000)	0.177, **0.284**, 0.233 (0.8)
ft7T	0.199, 0.263, 0.195 (0.5)	**0.236**, **0.283**, **0.213** (4000)	0.215, 0.271, 0.196 (0.8)
la7T	0.179, 0.238, 0.205 (0.4)	**0.220**, **0.294**, **0.233** (2000)	0.194, 0.268, 0.216 (0.8)
fbis8T	0.306, 0.344, 0.282 (0.01)	**0.334**, **0.367**, **0.292** (500)	0.319, 0.363, 0.288 (0.5)
ft8T	0.310, 0.359, 0.283 (0.3)	0.324, **0.367**, **0.297** (800)	**0.326**, **0.367**, 0.296 (0.7)
la8T	0.231, 0.264, 0.211 (0.2)	**0.258**, 0.271, 0.216 (500)	0.238, **0.282**, **0.224** (0.8)
trec7T	0.167, 0.366, 0.315 (0.3)	**0.186**, **0.412**, **0.342** (2000)	0.172, 0.382, 0.333 (0.7)
trec8T	0.239, 0.438, 0.378 (0.2)	**0.256**, 0.448, 0.398 (800)	0.245, **0.466**, **0.406** (0.6)
web8T	0.243, 0.348, 0.293 (0.01)	**0.294**, **0.448**, **0.374** (3000)	0.242, 0.370, 0.323 (0.7)
Avg.	0.227, 0.323, 0.265	**0.256**, **0.352**, **0.289**	0.236, 0.339, 0.279

Collection	Jelinek-Mercer avgpr, pr@10d, pr@20d (λ)	Dirichlet Prior avgpr, pr@10d, pr@20d (μ)	Absolute Discounting avgpr, pr@10d, pr@20d (δ)
fbis7L	0.224, **0.339**, **0.279** (0.7)	**0.232**, 0.313, 0.249 (5000)	0.185, 0.321, 0.259 (0.6)
ft7L	0.279, **0.331**, 0.244 (0.7)	**0.281**, 0.329, **0.248** (2000)	0.249, 0.317, 0.236 (0.8)
la7L	0.264, 0.350, **0.286** (0.7)	**0.265**, **0.354**, 0.285 (2000)	0.251, 0.340, 0.279 (0.7)
fbis8L	0.341, 0.349, 0.283 (0.5)	**0.347**, 0.349, **0.290** (2000)	0.343, **0.356**, 0.274 (0.7)
ft8L	**0.375**, **0.427**, **0.320** (0.8)	0.347, 0.380, 0.297 (2000)	0.351, 0.398, 0.309 (0.8)
la8L	**0.290**, **0.296**, **0.238** (0.7)	0.277, 0.282, 0.231 (500)	0.267, 0.287, 0.222 (0.6)
trec7L	0.222, **0.476**, **0.401** (0.8)	**0.224**, 0.456, 0.383 (3000)	0.204, 0.460, 0.396 (0.7)
trec8L	**0.265**, 0.504, **0.434** (0.8)	0.260, 0.484, 0.400 (2000)	0.248, **0.518**, 0.428 (0.8)
web8L	0.259, **0.422**, **0.348** (0.5)	**0.275**, 0.410, 0.343 (10000)	0.253, 0.414, 0.333 (0.6)
Avg.	**0.280**, **0.388**, **0.315**	0.279, 0.373, 0.303	0.261, 0.379, 0.304

Table 3: Comparison of smoothing methods on title queries (top) and long queries (bottom). The three numbers in each cell are average precision, precision at 10 documents, and precision at 20 documents. The parameter chosen for each data set and method is shown in parentheses.

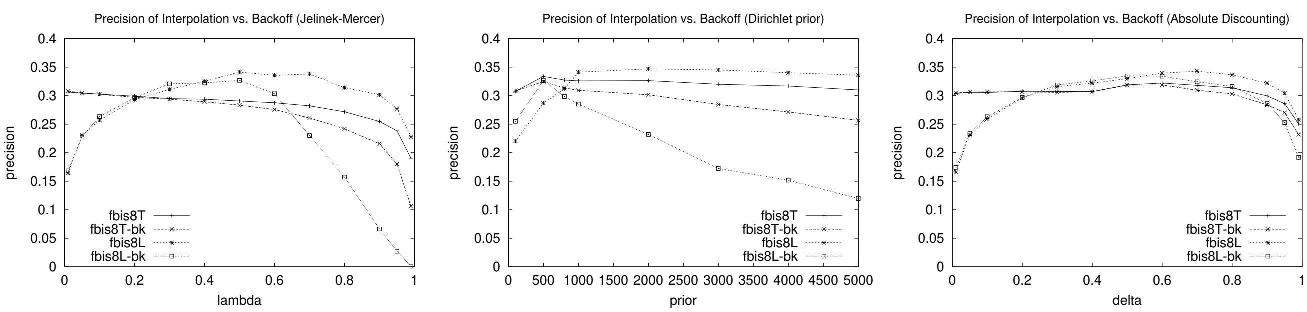

Figure 5: Interpolation versus backoff for Jelinek-Mercer (top), Dirichlet smoothing (middle), and absolute discounting (bottom).

the backoff version tends to do term weighting and document length normalization more aggressively than the corresponding interpolated version.

The backoff strategy and the interpolation strategy are compared for all three methods using the FBIS database and topics 401-450 (i.e., fbis8T and fbis8L). The results are shown in Figure 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 ($\frac{|d|_u}{|d|}$) 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.

8. CONCLUSIONS AND FUTURE WORK

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. An 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 long and verbose queries than for concise title queries. This suggests that smoothing may be playing two different roles in the query likelihood retrieval method. One role is to improve the accuracy of the estimated document language model, while the other is to accommodate generation of non-informative common words in the query. The results further suggest that Dirichlet prior may be good for the estimation role, while Jelinek-Mercer may be good for the query modeling role.

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

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.

There are several interesting future research directions that will help better understand the role of smoothing. First, it would be interesting to test with queries that are long, but non-verbose, or short but verbose. This will help clarify whether it is the length or the verbosity of the query that is interacting with the effect of smoothing. Second, one can de-couple the two different roles of smoothing by adopting a two stage smoothing strategy in which Dirichlet smoothing is first applied to implement the estimation role and Jelinek-Mercer smoothing is then applied to implement the role of query modeling. Finally, there are many other effective smoothing algorithms that we have not yet tested (e.g., Good-Turing smoothing [4], Katz smoothing [7], Kneser-Ney smoothing [8]); evaluation of them would be a natural further research direction. It is also very important to study how to exploit the past relevance judgments, the current query, and the current database to train the smoothing parameters, since, in practice, it would not be feasible to search the whole parameter space as we did in this paper.

ACKNOWLEDGEMENTS

We thank Jamie Callan, Bruce Croft, John Prange, and the three anonymous reviewers for helpful comments on this work. 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.

REFERENCES

[1] A. Berger and J. Lafferty (1999). "Information retrieval as statistical translation," In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 222–229.

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

[3] N. Fuhr (1992). "Probabilistic models in information retrieval", *The Computer Journal*, Vol.35, No.3, pp. 243–255.

[4] I. J. Good (1953). "The Population Frequencies of Species and the Estimation of Population Parameters," *Biometrika*, Volume 40, parts 3,4, pp. 237–264.

[5] D. Hiemstra and W. Kraaij (1998). "Twenty-one at TREC-7: Ad-hoc and cross-language track," in *Proc. of Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, MD.

[6] F. Jelinek and R. Mercer (1980). "Interpolated estimation of Markov source parameters from sparse data". In *Pattern Recognition in Practice*, E. S. Gelsema and L. N. Kanal (editors), pages 381–402. North Holland, Amsterdam.

[7] S. M. Katz (1987). "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, volume ASSP-35, pages 400–401, March 1987.

[8] R. Kneser and H. Ney (1995). "Improved smoothing for m-gram language modeling," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Detroit, MI.

[9] MacKay, D. and Peto, L. (1995). "A hierarchical Dirichlet language model." *Natural Language Engineering*, 1(3), pp. 289–307.

[10] D. H. Miller, T. Leek, and R. Schwartz (1999). "A hidden Markov model information retrieval system," In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 214–221.

[11] H. Ney, U. Essen, and R. Kneser (1994). "On structuring probabilistic dependencies in stochastic language modeling," *Computer Speech and Language*, 8:1-38.

[12] J. Ponte (1998). *A language modeling approach to information retrieval.* Ph.D. thesis, University of Massachusetts at Amherst.

[13] J. Ponte and W. B. Croft (1998). "A language modeling approach to information retrieval," *Proceedings of the ACM SIGIR*, pp. 275–281.

[14] C. J. van Rijsbergen (1986). "A Non-classical Logic for Information Retrieval," *The Computer Journal*, 29(6).

[15] S. E. Robertson, C. J. van-Rijsbergen, and M. F. Porter (1981). "Probabilistic models of indexing and searching", in Oddy R. N. et al. (Eds.) Information Retrieval Research, Butterworths, London, 1981, pp. 35–56.

[16] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford (1995). "Okapi at TREC-3," *The Third Text REtrieval Conference (TREC-3)*, in D. K. Harman (ed), NIST Special Publication.

[17] G. Salton and C. Buckley (1988). "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, **24**, pp. 513–523.

[18] G. Salton and C. Buckley (1990), "Improving retrieval performance by relevance feedback", Journal of the American Society for Information Science, Vol. 44, No. 4, 288–297.

[19] A. Singhal, C. Buckley, and M. Mitra (1996). "Pivoted document length normalization," in *Proceedings of the 1996 ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 21–29.

[20] F. Song and B. Croft (1999). "A general language model for information retrieval," in *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 279–280.

[21] K. Sparck Jones (1997). *Readings in Information Retrieval*, P. Willett, ed., Morgan Kaufmann Publishers.

[22] S. K. M. Wong and Y. Y. Yao (1995), "On modeling information retrieval with probabilistic inference," *ACM Transactions on Information Systems*, 13(1), pp. 69–99.