

# The Dual Role of Smoothing in the Language Modeling Approach

Chengxiang Zhai and John Lafferty

School of Computer Science  
Carnegie Mellon University  
{czhai, lafferty}@cs.cmu.edu

## Abstract

In this paper, we study the role of smoothing in the language modeling approach to text retrieval. We show that the involving of the collection language model in smoothing generally forces an implicit TF-IDF weighting in the retrieval formula. We empirically compare several different smoothing methods and examine the sensitivity of retrieval precision to the setting of smoothing parameters. Experiment results indicate that retrieval performance can be very sensitive to smoothing. The results also suggest that smoothing plays two quite different roles in the query likelihood ranking function. One role is to avoid assigning zero probabilities to words that have not occurred in a document and the other is to accommodate generation of common words in a query. We propose a two-stage smoothing strategy to explicitly de-couple these two roles of smoothing, and show that this strategy not only gives us better control over the smoothing parameters, but also often outperforms any of the single smoothing method.

## 1 The Language Modeling Approach

The language modeling approach was first proposed in [Ponte and Croft, 1998]. The basic idea is to rank documents based on the likelihood of the query being generated from a probabilistic model (language model) estimated based on each document. Although the probabilistic model actually used in [Ponte and Croft, 1998] is a multivariate Bernoulli model, its estimation is based on a unigram language model induced by a document. Other works later, such as [Hiemstra and Kraaij, 1998, Miller et al., 1999, Berger and Lafferty, 1999, Ng, 1999] motivate this basic approach from different perspectives and use different smoothing strategies, but the essential formula used for ranking documents are all of the same form of query likelihood, which is given below.

Let  $q = q_1 q_2 \dots q_n$  be a query and  $d = d_1 d_2 \dots d_m$  a document. The basic ranking formula is the query log-likelihood (assuming unigram language model)  $\log p(q|d) = \sum_i \log p(q_i|d)$ . Clearly, the retrieval problem is now essentially reduced to a unigram language model estimation problem, i.e., to estimate  $p(q_i|d)$ .

Smoothing of document language model is necessary for the obvious reason that the standard maximum likelihood estimator would assign a zero probability to all words that have not occurred in a document, thus one basic question that we can ask about this new approach is how smoothing may affect retrieval effectiveness. In this paper, we summarize the results from several attempts to understand the role of smoothing in the language modeling approach to retrieval. We present empirical results to show that smoothing is necessary not just to avoid assigning zero probabilities to words, but also to accommodate generation of common words in a query. We propose a two-stage smoothing strategy that explicitly de-couples these two different roles of smoothing. We show that this strategy not only gives us better control over the smoothing parameters, but also outperforms any of the single smoothing method.

## 2 Smoothing and Implicit TF-IDF Term Weighting

There is an interesting connection between the language model approach and the TFIDF weighting heuristics used in the traditional models. The connection has much to do with smoothing, and an appreciation of it helps to gain insight into the language modeling approach. In particular, when the probability of an unseen word  $w$  in document  $d$  ( $p_u(w|d)$ ) is assumed to be proportional to its probability according to the unigram language model estimated based on the whole document collection ( $p(w|\mathcal{C})$ ), i.e.,  $p_u(w|d) = \alpha_d p(w|\mathcal{C})$ , where  $\alpha_d$  is a document-dependent constant, the query likelihood retrieval formula can be written as

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

<i>Method</i>	$p_s(w   d)$	$\alpha_d$	<i>Parameter</i>
Jelinek-Mercer	$\lambda p_{ml}(w   d) + (1 - \lambda) p(w   \mathcal{C})$	$1 - \lambda$	$\lambda$
Dirichlet	$\frac{c(w; d) + \mu p(w   \mathcal{C})}{\sum_w c(w; d) + \mu}$	$\frac{\mu}{\sum_w c(w; d) + \mu}$	$\mu$
Absolute discount	$\frac{\max(c(w; d) - \delta, 0)}{\sum_w c(w; d)} + \frac{\delta  d _u}{ d } p_c(w)$	$\frac{\delta  d _u}{ d }$	$\delta$

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

where  $p_s(\cdot | d)$  is a smoothed probability for a word seen in  $d$ ,  $c(q_i; d)$  is the count of word  $q_i$  in  $d$ , and  $n$  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. See [Zhai and Lafferty, 2001] for details of this derivation.

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  $\frac{p_s(q_i | d)}{\alpha_d p(q_i | \mathcal{C})}$ , which is directly proportional to the document term frequency ( $p_s(q_i | d)$ ), but inversely proportional to the collection frequency ( $p(q_i | \mathcal{C})$ ).

Thus, the use of  $p(q_i | \mathcal{C})$  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 in the traditional model to improve performance. Indeed,  $\alpha_d$  should be closely related to the document length, since one would expect that a longer document needs less smoothing and thus a smaller  $\alpha_d$ ; thus a long document incurs a greater penalty than a short one because of this term.

The connection just derived shows that TF-IDF weighting heuristics and document length normalization can both be captured by the language model approach in terms of the smoothing of document language models, and indicates that smoothing plays a key role in such approaches. A similar derivation, but restricted to a special smoothing method, was given in [Hiemstra and Kraaij, 1998].

### 3 Comparison of Three Smoothing Methods

In this section, we present some empirical results from comparing three simple popular smoothing methods, and show that the retrieval precision is sensitive to the setting of smoothing parameters in some interesting ways. It suggests that smoothing plays two quite different roles in the language modeling approach. More details about these results can be found in [Zhai and Lafferty, 2001].

Following [Chen and Goodman, 1998], we assume the general form of a smoothed model to be the following:

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

where  $p_s(w | d)$  is the smoothed probability of a word seen in the document,  $p(w | \mathcal{C})$  is the collection language model, and  $\alpha_d$  is a coefficient controlling the probability mass assigned to unseen words, so that all probabilities sum to one. Thus, individual smoothing methods essentially differ only in their choice of  $p_s(w | d)$ . Note that, in general,  $\alpha_d$  may depend on  $d$ .

The three smoothing methods that we considered are summarized in Table 1 in terms of  $p_s(w | d)$  and  $\alpha_d$  in the general form. Informally, the *Jelinek-Mercer* method is simply a linear interpolation of the maximum likelihood model with the collection model, using a coefficient  $\lambda$  to control the influence of each model. The *Dirichlet prior* method essentially assumes that each word has some extra ‘‘prior counts’’ that are proportional to the probability of the word according to the collection language model, with  $\mu$  controlling the scale of the prior counts. The *absolute discounting* method is to discount a seen word by subtracting some constant ( $\delta$ ) from its count and re-allocate these extra counts according to the collection language model. Note that if we plug Jelinek-Mercer smoothing into the general query likelihood retrieval formula, we can see that there is always some IDF effect whenever the query is generated from a mixture model with one component being the collection model. We will exploit this property in our two-stage smoothing strategy.

The three methods are tested on several TREC collections. Some major observations are:

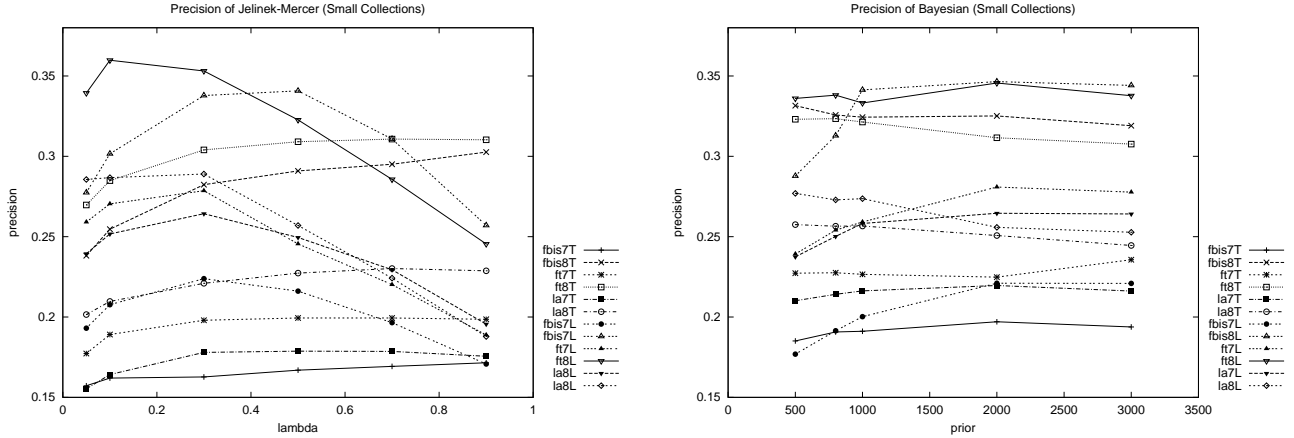


Figure 1: Precision of Jelinek-Mercer smoothing (left) and Dirichlet smoothing (right) at different parameter values.

1. The performance of long and verbose queries is much more sensitive to the setting of the smoothing parameters than that of title queries.
2. The optimal setting of smoothing parameters is sensitive to the types/lengths of the query.
3. Dirichlet prior tends to work very well on title queries (consistently better than the other two), but not as well on long queries.
4. Jelinek-Mercer generally benefits much more from long queries than the other two approaches do.

For example, Figure 1 shows the non-interpolated average precision at different values of the smoothing parameter in the Jelinek-Mercer method and the Dirichlet prior method for several different testing collections. The label ending with an “L” (or “T”) refers to a run with long (or title) queries. It is easy to see that the “T” runs are generally flatter than the “L” runs, indicating less sensitivity for title queries than for long queries. We also see the trend that the optimal  $\lambda$  in Jelinek-Mercer smoothing is generally above 0.9 for title queries but around 0.3 for long queries.

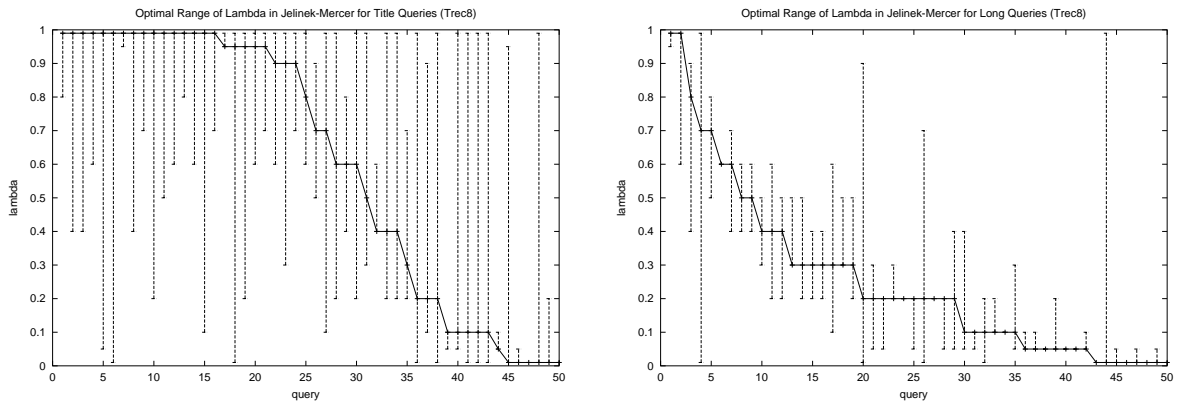


Figure 2: Optimal lambda range for title queries (left) and long queries (right).

Such sensitivity and trend can be seen more clearly from the per-topic plot of the optimal range of  $\lambda$  shown in Figure 2. The optimal range is defined as the maximum range of  $\lambda$  values that deviate from the optimal average precision by no more than 0.01. The line in the picture shows the optimal value of  $\lambda$  and the bars are the optimal ranges.

If the only purpose of smoothing is to avoid assigning zero probabilities, then we should expect the optimal setting of smoothing parameters for a document language model to be more related to the properties of the

document collection and relatively insensitive to the difference in queries. For example, we may expect the optimal setting to be quite similar on the same collection for all kinds of queries.

However, the results above show that it is not the case: The optimal setting of smoothing parameters is actually quite sensitive to the query – different queries prefer a different optimal setting and in general title queries prefer less smoothing, while long verbose queries prefer more smoothing. This suggests that smoothing may be playing a double role in the query likelihood retrieval approach: (1) It avoids assigning a zero probability to a word not seen in a document. This role has been well-recognized, and is, indeed, the primary motivation for smoothing at the first place. (2) It “helps” generate common words in the query, so as to causes query terms to be weighted in a similar fashion as the IDF heuristics. This role can be understood from the analysis in Section 2. It is likely that the effect we have seen is actually a *mixed* effect of both roles. For title or keyword queries, we should expect the second role to be much less significant than the first one, so the overall effect is largely due to the first role. For long and verbose queries, we may expect the second role to be much more significant since there are more common and non-informative words to account for. Thus the overall effect of smoothing for long queries is more likely a mixture, i.e., due to both roles.

The dual-role hypothesis above suggests a possible explanation for the somehow “unexpected” results we discussed in the previous section: It is possible that the Dirichlet prior method is best for the first role of smoothing, since it adapts to document length as it should be, while Jelinek-Mercer is best for the second role, since it uses a fixed coefficient across all documents, so is equivalent to using a simple generative mixture model for queries. For title queries, since the query term weighting role is insignificant, the overall performance is dominated by the effect of the first role, thus Dirichlet prior outperforms all other methods. For long queries, we observed a mixed effect. Jelinek-Mercer performs well because it plays the second role well, while Dirichlet prior also performs well because it plays the first role well. Jelinek-Mercer benefits most from long queries, because they give it a chance to show the superiority in playing the second role of smoothing. Dirichlet prior has not gained much from long queries, because it is relatively ineffective when playing the second role.

## 4 Two-Stage Smoothing

The analysis in the previous section suggests the following two-stage smoothing strategy:

At the first stage, a document language model is smoothed using Dirichlet prior (to implement the first role); At the second stage, it is further smoothed using Jelinek-Mercer (to implement the second role). Since Jelinek-Mercer smoothing can be regarded as a simple mixture model, the second stage is essentially to introduce a generative collection mixture model for queries. In this sense, the two-stage smoothing strategy is similar to the two-state HMM used in [Miller et al., 1999], but there was no first-stage smoothing in their HMM.

The two-stage smoothing strategy can also be understood simply as a combination of Dirichlet prior smoothing with Jelinek-Mercer smoothing. The combined smoothing function is given by

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

where,  $\lambda$  is the Jelinek-Mercer smoothing parameter and  $\mu$  is the Dirichlet prior parameter.

Plugging this smoothing function into the general query likelihood ranking formula in Section 2, dropping the document-independent constant, we have the following ranking formula with two stage smoothing:

$$\log p(q|d) = \sum_{i:c(q_i;d)>0} \log\left(1 + \frac{\lambda c(q_i;d)}{((1-\lambda)|d| + \mu)p(q_i|\mathcal{C})}\right) + n \log\left(\frac{(1-\lambda)|d| + \mu}{|d| + \mu}\right)$$

It is easy to verify that when  $\lambda = 1$ , we end up with just the Dirichlet prior smoothing.

On the surface, we now have two, rather than one, parameter to worry about. However, the two parameters  $\lambda$  and  $\mu$  are now more meaningful.  $\lambda$  is intended to be a query-related parameter (even though it is not tightly coupled with the query term frequency in the retrieval formula!) and should be smaller if the query is more verbose and long. It can be roughly interpreted as modeling the “purity” of the query.  $\mu$  is a document (collection) related parameter, and controls the amount of probability mass assigned to unseen words. Given a document collection, the optimal value of  $\mu$  is expected to be relatively stable. We now present some empirical results to show that this is indeed true.

Figure 3 shows how the precision varies when we change the prior value in Dirichlet smoothing. The three figures are for three different testing collections respectively (Trec7 ad hoc task, Trec8 ad hoc task, and Trec8 web task). 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,  $\lambda = 0.3$ ) with long

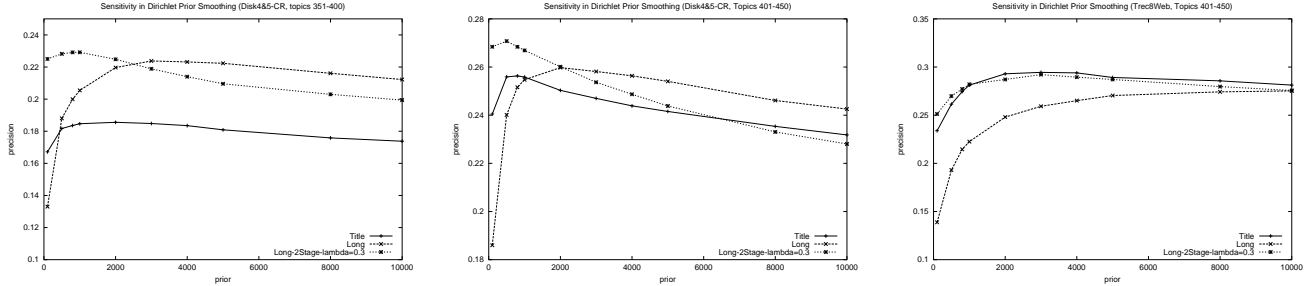


Figure 3: Sensitivity of precision on Trec7 (left), Trec8 (middle), and web (right) testing collection.

queries<sup>1</sup>. (1) and (2) can be interpreted as two-stage smoothing with  $\lambda = 1$ , or approximately as two-stage smoothing with optimal  $\lambda$ . (The two-stage results with true optimal  $\lambda$  is almost the same as the results shown here.)

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 more similar for title queries and long queries, i.e., it becomes much less sensitive to the type/length of queries. By factoring out the second role of smoothing, we 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 (left) 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*  $\lambda$  value can stabilize the sensitivity pattern of Dirichlet smoothing on *different* collections. The Trec8 figure (middle) and web figure (right) involve the same topic set but different collections. By comparing them, we notice that the same  $\lambda$  value (0.3) 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 (right) and notice that the position of optimal prior is different.

We should also notice, from these figures, that with the help from Jelinek-Mercer in two-stage smoothing, Dirichlet can achieve a higher precision on long queries than it could when we just do Dirichlet smoothing alone. Of course, it could have been that Jelinek-Mercer is just a better method for long queries, but the two stage results are also better than the best result of using Jelinek-Mercer alone (see Table 4).

Collection	Best Jelinek-Mercer	Best Dirichlet	Best Two-Stage
Trec7-long	0.221	0.222	0.229
Trec8-long	0.261	0.255	0.271
Web8-long	0.260	0.274	0.292

Table 2: Average precision of two-stage smoothing and one-stage smoothing

It is worth mentioning that, on the web data (Figure 3, right), 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 3, 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.

<sup>1</sup>0.3 is the optimal value of  $\lambda$  when Jelinek-Mercer method is applied alone

## 5 Conclusions

In this paper, we studied the role of smoothing in the language modeling approach. We show analytically that the use of a language model estimated based on the whole document collection as a reference model for smoothing implies a retrieval formula that implements similar heuristics like TF-IDF term weighting and document length normalization. This suggests that smoothing is indeed playing a very important role in this new approach to retrieval and retrieval performance will be sensitive to the choice of smoothing method and setting of parameters. Evaluation of three different smoothing methods on several collections confirmed such sensitivity. The somehow unexpected observation that the optimal setting of smoothing parameters is quite sensitive to the type or length of the query suggests that smoothing plays two quite different roles in the query likelihood retrieval function. One role is to avoid assigning zero probabilities to unseen words in a document and the other is to accommodate generation of common words in a query. This hypothesis provides a possible explanation for the seemingly unusual patterns occurring in our results. In particular, it suggests that the Dirichlet prior method is best to play the first role, while the Jelinek-Mercer method is best to play the second role. This then motivates us to propose a two-stage smoothing strategy that combines Dirichlet prior with Jelinek-Mercer and explicitly de-couples the two roles of smoothing (Dirichlet prior for first role and Jelinek-Mercer for the second). Experiments with the two-stage smoothing method show that it indeed results in a more meaningful pattern of the smoothing parameters. Specifically, with two-stage smoothing, the optimal setting of Jelinek-Mercer  $\lambda$  is more correlated with the query while that of Dirichlet prior  $\mu$  is more related to the document collection. The two-stage smoothing method also consistently outperforms any of the single smoothing method, even though only slightly sometimes. We believe that the two-stage smoothing can serve as a reasonable baseline approach for testing new retrieval algorithms with language modeling.

A natural future direction is to study how to set the two parameters in the two-stage method. In all the reported experiments, we exhaustively search the space of the parameter values. In practice, this would be impossible, and we need to have guidance on how to set them. It would be very interesting to explore the possibility of training these parameters using the query, the collection, as well as the past relevance judgments that are available to us.

## References

- [Berger and Lafferty, 1999] Berger, A. and Lafferty, J. (1999). Information retrieval as statistical translation. In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229.
- [Chen and Goodman, 1998] Chen, S. F. and Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.
- [Hiemstra and Kraaij, 1998] Hiemstra, D. and Kraaij, W. (1998). Twenty-one attrec-7: Ad-hoc and cross-language track. In *Proc. of Seventh Text REtrieval Conference (TREC-7)*.
- [Miller et al., 1999] Miller, D. H., Leek, T., and Schwartz, R. (1999). A hidden markov model information retrieval system. In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214–221.
- [Ng, 1999] Ng, K. (1999). A maximum likelihood ratio information retrieval model. In *Proc. of Eighth Text REtrieval Conference (TREC-8)*.
- [Ponte and Croft, 1998] Ponte, J. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR*, pages 275–281.
- [Zhai and Lafferty, 2001] Zhai, C. and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'2001*. To appear.