Two-Stage Language Models for 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

The optimal settings of retrieval parameters often depend on both the document collection and the query, and are usually found through empirical tuning. In this paper, we propose a family of *two-stage* language models for information retrieval that explicitly captures the different influences of the query and document collection on the optimal settings of retrieval parameters. As a special case, we present a two-stage smoothing method that allows us to estimate the smoothing parameters completely automatically. In the first stage, the document language model is smoothed using a Dirichlet prior with the collection language model as the reference model. In the second stage, the smoothed document language model is further interpolated with a query background language model. We propose a leave-one-out method for estimating the Dirichlet parameter of the first stage, and the use of document mixture models for estimating the interpolation parameter of the second stage. 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 Search and Retrieval**]: Retrieval Models—*language models, parameter setting*

General Terms

Algorithms

Keywords

risk minimization, two-stage language models, two-stage smoothing, Dirichlet prior, interpolation, parameter estimation, leave-one-out, mixture model

1. INTRODUCTION

It is well-known that the optimal settings of retrieval parameters generally depend on both the document collection and the query. For example, specialized term weighting for short queries was studied in [3]. Salton and Buckley studied many different term weighting methods used in the vector-space retrieval model; their recommended methods strongly depend on the type of the query and the characteristics of the document collection [13]. 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 [15].

The need for empirical parameter tuning is due in part from the fact that most existing retrieval models are based on certain *pre-assumed* representation of queries and documents, rather than on a direct modeling of the queries and documents. As a result, the "adaptability" of the model is restricted by the particular representation assumed, and reserving free parameters for tuning becomes a way to accommodate any difference among queries and documents that has not been captured well in the representation. In order to be able to set parameters automatically, it is necessary to model queries and documents directly. This goal has been explored recently in the language modeling approach to information retrieval, which has attracted significant attention since it was first proposed in [9].

The first uses of the language modeling approach focused on its empirical effectiveness using simple models [9, 7, 2, 1]. Recent work has begun to develop more sophisticated models and a systematic framework for this new family of retrieval methods. In [4], a risk minimization retrieval framework is proposed that incorporates language modeling as natural components, and that unifies several existing retrieval models in a framework based on Bayesian decision theory. One important advantage of the risk minimization retrieval framework over the traditional models is its capability of modeling both queries and documents directly through statistical language models, which provides a basis for exploiting statistical estimation methods to set retrieval parameters automatically. Several special language models are explored in [6, 4, 16], and in all uses of language modeling in IR, smoothing plays a crucial role. The empirical study in [17] reveals that not only is retrieval performance generally sensitive to the setting of smoothing parameters, but also that this sensitivity depends on the type of queries that are input to the system.

In this paper, we propose a family of language models for information retrieval that we refer to as *two-stage models*. The first stage involves the estimation of a *document* language model independent of the query, while the second stage involves the computation of the likelihood of the query according to a *query* language model,

SIGIR'02, August 11-15, 2002, Tampere, Finland.

which is based on the estimated document language model. Thus, the two-stage strategy explicitly captures the different influences of the query and document collection on the optimal settings of retrieval parameters.

We derive the two-stage models within the general risk minimization retrieval framework, and present a special case that leads to a two-stage *smoothing* method. In the first stage of smoothing, the document language model is smoothed using a Dirichlet prior with the collection language model as the reference model. In the second stage, the smoothed document language model is further interpolated with a query background language model. 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. Evaluation on five different databases and four types of queries indicates that the two-stage smoothing method with the proposed parameter estimation method—which is fully automatic—consistently gives retrieval performance that is close to, or better than, the result of using a single smoothing method and exhaustive parameter search on the test data.

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

The rest of the paper is organized as follows. We first derive the two-stage language models in Section 2, and present the two-stage smoothing method as a special case in Section 3. We then describe, in Section 4, methods for estimating the two parameters involved in the two-stage smoothing method. We report our experimental results in Section 5. Section 6 presents conclusions and suggestions for future work.

2. TWO-STAGE LANGUAGE MODELS

2.1 The risk minimization framework

The risk minimization retrieval framework is a general probabilistic retrieval framework based on Bayesian decision theory [4]. 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. The framework unifies several existing retrieval models within one general probabilistic framework, and facilitates the development of new principled approaches to text retrieval.

In traditional retrieval models, such as the vector-space model [12] and the BM25 retrieval model [11], 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 [14]). 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 [11].

One important advantage of the risk minimization retrieval framework [4] over these traditional models is its capability of modeling both queries and documents directly through statistical language modeling. 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. In general, using statistical language models allows us to introduce all parameters in a probabilistic way, and also makes it possible to set the parameters automatically through statistical estimation methods.

2.2 Derivation of two-stage language models

The original language modeling approach as proposed in [9] 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 formally using the risk minimization framework.

In the risk minimization framework presented in [4], documents are ranked based on the following risk function:

$$\begin{aligned} R(\mathbf{d}_i;\mathbf{q}) &= \sum_{R\in\{0,1\}} \int_{\Theta_Q}\int_{\Theta_D} L(\theta_Q,\theta_D,R)\ \times \\ & \qquad p(\theta_Q\,|\,\mathbf{q},\mathcal{U})\,p(\theta_D\,|\,d_i,\mathcal{S})\,p(R\,|\,\theta_Q,\theta_D)\,d\theta_D\,d\theta_Q \end{aligned}$$

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

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

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

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

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

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

$$R(\mathbf{d};\mathbf{q}) \sim -p(\hat{\theta}_D\,|\,\mathbf{q},\mathcal{U})$$

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

$$p(\hat{\theta}_D \,|\, \mathbf{q}, \mathcal{U}) \;\propto\; p(\mathbf{q} \,|\, \hat{\theta}_D, \mathcal{U}) p(\hat{\theta}_D \,|\, \mathcal{U}) \tag{1}$$

Equation 1 is our basic two-stage language model retrieval formula. Similar to the model discussed in [1], this formula has the following interpretation: $p(\mathbf{q} \,|\, \hat{\theta}_D, \mathcal{U})$ 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 $p(\mathbf{d} \,|\, \theta_D, \mathcal{S})$ for generating documents and a concrete model $p(\mathbf{q} \,|\, \theta_Q, \mathcal{U})$ 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 [9]; that is, we first estimate a document language model and then compute the query likelihood using the estimated model. In the next section, we present the generative models that lead to the two-stage smoothing method suggested in [17].

3. THE TWO-STAGE SMOOTHING METHOD

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

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

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

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

$$\mathrm{Dir}\,(\theta \,|\, \alpha) \;=\; \frac{\Gamma(\sum_{i=1}^{|V|} \alpha_i)}{\prod_{i=1}^{|V|} \Gamma(\alpha_i)} \prod_{i=1}^{|V|} \theta_i^{\alpha_i - 1} \tag{2}$$

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

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

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

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

where $|\mathbf{d}| = \sum_{w \in V} c(w, \mathbf{d})$ is the length of $\mathbf{d}$. This is the Dirichlet prior smoothing method described in [17].

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 [17], such a model will not be able to explain the interactions between smoothing and the type of queries. In order to capture the common and non-discriminative words in a query, we assume that a query is generated by sampling words from a two-component mixture of multinomials, with one component being θ_Q and the other some query background language model $p(\cdot \,|\, \mathcal{U})$. That is,

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

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

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

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

In this formula, the document language model is effectively smoothed in two steps. First, it is smoothed with a Dirichlet prior, and second, it is interpolated with a query background model. Thus, we refer to this as *two-stage smoothing*.

The above model has been empirically motivated by the observation that smoothing plays two different roles in the query likelihood retrieval method. One role is to improve the maximum likelihood estimate of the document language model, at the very least assigning non-zero probabilities to words that are not observed in the document. The other role is to "explain away" the common/non-discriminative words in the query, so that the documents will be discriminated primarily based on their predictions of the "topical" words in the query. The two-stage smoothing method explicitly decouples these two roles. The first stage uses Dirichlet prior smoothing method to improve the estimate of a document language model; this method normalizes documents of different lengths appropriately with a prior sample size parameter, and performs well empirically [17]. The second stage is intended to bring in a query background language model to explicitly accommodate the generation of common words in queries.

The query background model $p(\cdot \,|\, \mathcal{U})$ is in general different from the collection language model $p(\cdot \,|\, \mathcal{S})$. With insufficient data to estimate $p(\cdot \,|\, \mathcal{U})$, however, we can assume that $p(\cdot \,|\, \mathcal{S})$ would be a reasonable approximation of $p(\cdot \,|\, \mathcal{U})$. In this form, the two-stage smoothing method is essentially a combination of Dirichlet prior smoothing with Jelinek-Mercer smoothing [17]. Indeed, it is very easy to verify that when $\lambda = 0$, we end up with just the Dirichlet prior smoothing, whereas when $\mu = 0$, we will have Jelinek-Mercer smoothing. Since the combined smoothing formula still follows the general smoothing scheme discussed in [17], it can be implemented very efficiently. In the next section, we present methods for estimating μ and λ from data.

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

Table 1: Estimated values of μ along with database characteristics.

4. PARAMETER ESTIMATION

4.1 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 non-zero probabilities to words not observed in a document; this is the usual role of language model smoothing. A useful objective function for estimating smoothing parameters is the "leave-one-out" likelihood, that is, the sum of the log-likelihoods of each word in the observed data computed in terms of a model constructed based on the data with the target word excluded ("left out"). This criterion is essentially based on cross-validation, and has been used to derive several well-known smoothing methods including the Good-Turing method [8].

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

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

Thus, our estimate of μ is

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

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

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

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

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

and

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

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 1. There is no clear correlation between the database characteristics shown in the table and the estimated value of μ.

4.2 Estimating λ

With the query model hidden, the query likelihood is

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

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

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

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

Thus, we assume that the query is generated from a mixture of N document models with unknown mixing weights $\{\pi_i\}_{i=1}^N$. 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 on documents that predict the query well in our likelihood function; presumably, these documents are also more likely to be relevant.

With this likelihood function, the parameters λ and $\{\pi_i\}_{i=1}^N$ can be estimated using the EM algorithm. The update formulas are

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

and

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

5. EXPERIMENTS

In this section we first present experimental results that confirm the dual-role of smoothing, which provides an empirical justification for using the two-stage smoothing method for retrieval. We then present results of the two-stage smoothing method using the estimated parameters, comparing it to the optimal performance from using single smoothing methods and an exhaustive parameter search.

5.1 Influence of Query Length and Verbosity on Smoothing

In [17], strong interactions between smoothing and the type of queries have been observed. However, it is unclear whether the high sensitivity observed on long queries is due to a higher density of common words in such queries, or to just the length. The two-stage smoothing method assumes that it is the former. In order to clarify this, we design experiments to examine two query factors—length and "verbosity." Specifically, we consider four different types of queries, i.e., short keyword, long keyword, short verbose, and long verbose queries, and compare how they each behave with respect to smoothing. As we will show, the high sensitivity is indeed caused by the presence of common words in the query, and this provides an empirical justification for the two-stage smoothing method.

We generate the four types of queries 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 1 shows an example of such a topic (topic 52).

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

Concepts:

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

Figure 1: Example topic, number 52. The keywords are used as the "long keyword" version of our queries.

We use all of the 150 topics, and generate the four versions of queries in the following way:

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

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

The relevance judgments available for these 150 topics are mostly on the documents in TREC disk 1 and disk 2. In order to observe any possible difference in smoothing caused by the types of documents, we partition the documents in disks 1 and 2 and use the three largest subsets of documents, accounting for a majority of the relevant documents for our queries. The three databases are AP88-89, WSJ87-92, and ZIFF1-2, each about 400MB–500MB in size. The queries without relevance judgments for a particular database were ignored for all of the experiments on that database. Four queries do not have judgments on AP88-89, and 49 queries do not have judgments on ZIFF1-2. Preprocessing of the documents is minimized; only a Porter stemmer is used, and no stop words are removed. Combining the four types of queries with the three databases gives us a total of 12 different testing collections.

To understand the interaction between different query factors and smoothing, we examine 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 2. In each case, we show how the average precision varies according to different values of the smoothing parameter.

From these figures, we see that the two types of keyword queries behave similarly, as do the two types of verbose queries. The retrieval performance is generally much less sensitive to smoothing in the case of the keyword queries than for the verbose queries, whether long or short. Therefore, the sensitivity is much more correlated with the verbosity of the query than with the length of the query. Indeed, the short verbose queries are clearly more sensitive than the long keyword queries. In all cases, insufficient smoothing is much more harmful for verbose queries than for keyword queries. This confirms that smoothing is indeed responsible for "explaining" the common words in a query, and provides an empirical justification for the two-stage smoothing approach.

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.

5.2 Effectiveness of the Two-stage Smoothing Method

To evaluate the two-stage smoothing method, we first test it on the same 12 testing collections as described earlier. These collections represent a very good diversity in the types of queries, but the databases are all homogeneous and relatively small. In order to further test the robustness of the two-stage smoothing method, we then test it on three much bigger and more heterogeneous TREC collections. These are the official ad hoc retrieval collections used in TREC-7, TREC-8, and the TREC-8 small web track. The official TREC-7 and TREC-8 ad hoc tasks have used the same document database (i.e., TREC disk4 and disk5 excluding the Congressional Record data), but different topics (topics 351-400 for TREC-7 and 401-450 for TREC-8). The TREC-8 web track and the TREC-8 official ad hoc task share the same 50 topics. Since these topics do not have a concept field, we have only three types of queries: short-keyword, short-verbose, and long-verbose. The size of these large collections is about 2GB in the original source. Again, we perform minimum pre-processing – only a Porter stemmer was used, and no

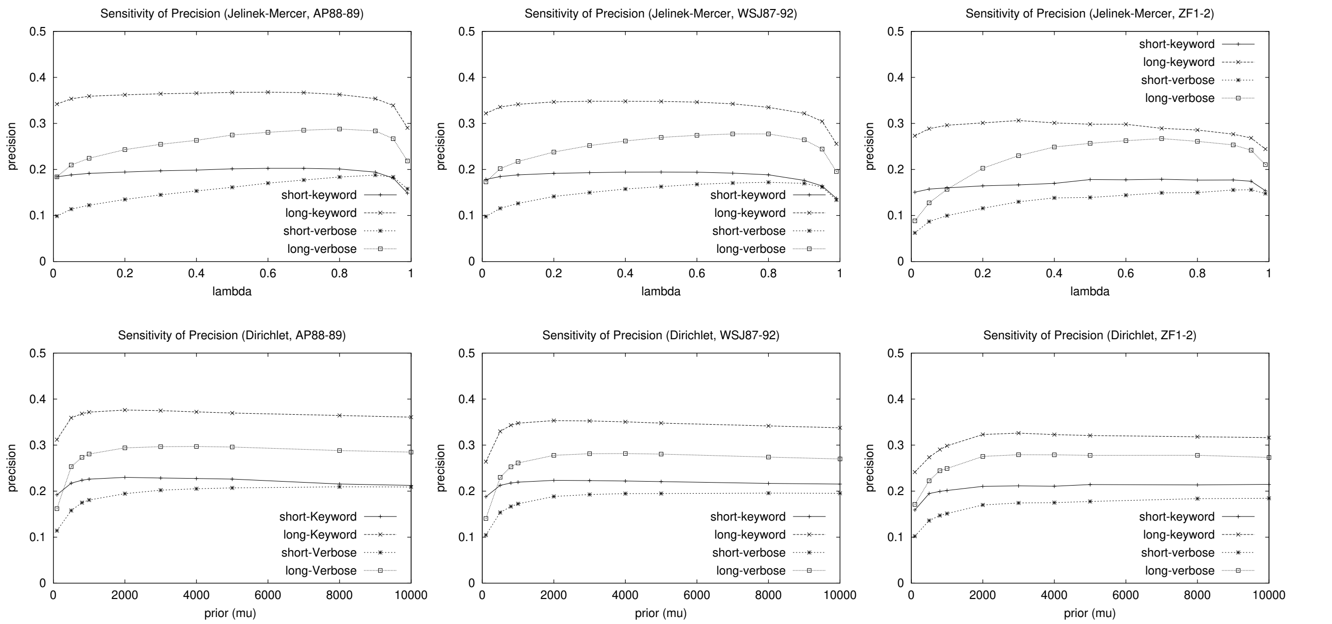

Figure 2: Sensitivity of Precision for Jelinek-Mercer smoothing (top) and Dirichlet prior smoothing (bottom) on AP88-89 (left), WSJ87-92 (center), and Ziff1-2 (right).

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 use the collection language model to approximate the query background model.

The results are shown in Table 2. The four types of queries are abbreviated with the two initial letters (e.g., SK for Short-Keyword). The standard TREC evaluation procedure for ad hoc retrieval is followed, and we have considered three performance measures – non-interpolated average precision, initial precision (i.e., precision at 0.0 recall), and precision at five 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 the three measures. Only in a few cases, the difference is statistically significant (indicated with a star).

To quantify the sensitivity of the retrieval performance to the smoothing parameter for single smoothing methods, we also show (in parentheses) the median average precision at all the parameter values that are tried[2]. 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 2, we see that as long as we set a relatively large value for μ in Dirichlet prior, the performance will not be much worse than the best performance, and there is a great chance that the median is 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 2, that such a simple approach would not perform so well as our parameter estimation methods. Indeed, the two-stage performance is always better than the median, except for three cases of short-keyword queries when it is slightly worse. Since the Dirichlet prior smoothing dominates the two-stage smoothing effect for these short-keyword queries (due to "little noise"), this somehow suggests that the leave-one-out method might have under-estimated 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 [4, 6, 16]. 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 deter-

[2]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}.

Database	*Query*	*Best Jelinek-Mercer*				*Best Dirichlet*				*Two-Stage*		
		AvgPr	(med)	InitPr	Pr@5d	AvgPr	(med)	InitPr	Pr@5d	AvgPr	Initpr	Pr@5d
AP88-89	SK	0.203	(0.194)	0.573	0.341	**0.230**	(0.224)	**0.623**	**0.381**	0.222*	0.611	0.375
	LK	0.368	(0.362)	**0.767**	0.530	**0.376**	(0.368)	0.755	**0.533**	0.374	0.754	**0.533**
	SV	0.188	(0.158)	0.569	0.342	**0.209**	(0.195)	**0.609**	**0.379**	0.204	0.598	0.368
	LV	0.288	(0.263)	**0.711**	0.463	**0.298**	(0.285)	0.704	**0.490**	0.292	0.689	0.473
WSJ87-92	SK	0.194	(0.188)	0.629	0.392	**0.223**	(0.218)	0.660	0.438	0.218*	**0.662**	**0.450**
	LK	0.348	(0.341)	0.814	0.597	0.353	(0.343)	0.834	**0.608**	**0.358**	**0.850***	0.607
	SV	0.172	(0.158)	0.615	0.377	0.196	(0.188)	0.638	0.413	**0.199**	**0.660**	**0.425**
	LV	0.277	(0.252)	**0.768**	**0.533**	0.282	(0.270)	0.750	0.504	**0.288***	0.762	0.520
ZF1-2	SK	0.179	(0.170)	0.455	0.248	**0.215**	(0.210)	**0.514**	**0.301**	0.200	0.494	0.299
	LK	0.306	(0.290)	0.675	0.404	**0.326**	(0.316)	0.681	**0.438**	0.322	**0.696**	0.434
	SV	0.156	(0.139)	0.450	0.224	**0.185**	(0.170)	0.456	0.255	0.181	**0.487**	**0.271***
	LV	0.267	(0.242)	0.593	0.339	**0.279**	(0.273)	0.606	0.378	**0.279***	**0.618**	**0.384**

Database	*Query*	*Best Jelinek-Mercer*				*Best Dirichlet*				*Two-Stage*		
		AvgPr	(med)	InitPr	Pr@5d	AvgPr	(med)	InitPr	Pr@5d	AvgPr	Initpr	Pr@5d
Trec7/8	Trec7-SK	0.167	(0.165)	0.632	0.400	**0.186**	(0.182)	**0.688**	**0.432**	0.182	0.673	0.420
	Trec7-SV	0.173	(0.138)	0.646	0.416	**0.182**	(0.168)	**0.656**	**0.436**	0.181	0.655	0.416
	Trec7-LV	0.222	(0.195)	0.723	0.496	0.224	(0.212)	**0.763**	**0.524**	**0.230**	0.760	0.516
	Trec8-SK	0.239	(0.237)	0.621	0.44	0.256	(0.244)	0.717	0.488	**0.257**	**0.719**	**0.496**
	Trec8-SV	**0.231**	(0.192)	0.687	0.456	0.228	(0.222)	0.670	0.432	**0.231**	**0.719**	**0.484**
	Trec8-LV	0.265	(0.234)	**0.789**	**0.556**	0.260	(0.252)	0.753	0.492	**0.268**	0.787	0.524
Web	Trec8-SK	0.243	(0.212)	0.607	0.368	**0.294**	(0.281)	**0.756**	0.484	0.278*	0.730	**0.488**
	Trec8-SV	0.203	(0.191)	0.611	0.392	**0.267**	(0.249)	**0.699**	**0.492**	0.253	0.680	0.436
	Trec8-LV	0.259	(0.234)	**0.790**	0.464	0.275	(0.248)	0.752	**0.508**	**0.284**	0.781	**0.508**

Table 2: Comparison of the estimated two-stage smoothing with the best single stage smoothing methods on small collections (top) and large collections (bottom). 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 single smoothing performance is statistically significant according to the Wilcoxin signed rank test at the level of 0.05.

mined 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.

6. CONCLUSIONS

In this paper we derive general two-stage language models for information retrieval using the risk minimization retrieval framework, and present a concrete two-stage smoothing method as a special case. The two-stage smoothing strategy explicitly captures the different influences of the query and document collection on the optimal settings of smoothing parameters. In the first stage, the document language model is smoothed using a Dirichlet prior with the collection model as the reference model. In the second stage, the smoothed document language model is further interpolated with a query background model.

We propose a leave-one-out method for estimating the first-stage Dirichlet prior parameter and a mixture model for estimating the second-stage interpolation parameter. These methods allow us to set the retrieval parameters automatically, yet adaptively according to different databases and queries. Evaluation on five different databases and four types of queries indicates that the two-stage smoothing method with the proposed parameter estimation scheme consistently gives retrieval performance that is close to, or better than, the best results attainable using a 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.

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. For example, it would be important to see the relative optimality of the estimated μ and λ when fixing one of them. 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. Another interesting future direction is to exploit the query background model to address the issue of redundancy in the retrieval results. Specifically, a biased query background model may be used to represent/explain the sub-topics that a user has already encountered (e.g., through reading previously retrieved results), in order to focus ranking on the *new* sub-topics in a relevant set of documents.

ACKNOWLEDGEMENTS

We thank Rong Jin, Jamie Callan, and the anonymous reviewers for helpful comments on this work. This research was sponsored in full 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.

REFERENCES

[1] 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.

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

[3] Kwok, K. and Chan, M. (98). Improving two-stage ad-hoc retrieval for short queries. In *Proceedings of SIGIR'98*, pages 250–256.

[4] Lafferty, J. and Zhai, C. (2001a). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'2001*.

[5] Lafferty, J. and Zhai, C. (2001b). Probabilistic IR models based on query and document generation. In *Proceedings of the Language Modeling and IR workshop*. Extended abstract.

[6] Lavrenko, V. and Croft, B. (2001). Relevance-based language models. In *Proceedings of SIGIR'2001*.

[7] 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.

[8] Ney, H., Essen, U., and Kneser, R. (1995). On the estimation of 'small' probabilities by leaving-one-out. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (12):1202–1212.

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

[10] Robertson, S. and Sparck Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146.

[11] Robertson, S. E., Walker, S., Jones, S., M.Hancock-Beaulieu, M., and Gatford, M. (1995). Okapi at TREC-3. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*.

[12] Salton, G., Wong, A., and Yang, C. S. (1975a). A vector space model for automatic indexing. *Communications of the ACM*, (11):613–620.

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

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

[15] Voorhees, E. and Harman, D., editors (2001). *Proceedings of Text REtrieval Conference (TREC1-9)*. NIST Special Publications. http://trec.nist.gov/pubs.html.

[16] Zhai, C. and Lafferty, J. (2001a). Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*.

[17] Zhai, C. and Lafferty, J. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'2001*.