Regularized Estimation of Mixture Models for Robust Pseudo-Relevance Feedback

Tao Tao
Department of Computer Science
University of Illinois at Urbana-Champaign

ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

ABSTRACT

Pseudo-relevance feedback has proven to be an effective strategy for improving retrieval accuracy in all retrieval models. However the performance of existing pseudo feedback methods is often affected significantly by some parameters, such as the number of feedback documents to use and the relative weight of original query terms; these parameters generally have to be set by trial-and-error without any guidance. In this paper, we present a more robust method for pseudo feedback based on statistical language models. Our main idea is to integrate the original query with feedback documents in a single probabilistic mixture model and regularize the estimation of the language model parameters in the model so that the information in the feedback documents can be gradually added to the original query. Unlike most existing feedback methods, our new method has no parameter to tune. Experiment results on two representative data sets show that the new method is significantly more robust than a state-of-the-art baseline language modeling approach for feedback with comparable or better retrieval accuracy.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Information retrieval, algorithm

Keywords

Pseudo feedback, mixture model, regulation, EM

1. INTRODUCTION

Among many techniques for improving the accuracy of ad hoc information retrieval, pseudo feedback is arguably the most effective one and has been shown to be effective across all retrieval models. The basic idea of pseudo feedback is to assume a certain number of top-ranked documents are relevant and learn from these documents to improve the query representation, thus to improve retrieval accuracy [1, 16]. Intuitively, since the top ranked documents are likely relevant or at least similar to relevant documents, we may exploit these documents to identify useful additional terms to improve the query representation and/or to assign better weights to the query terms. Such intuition has been implemented in different ways in different retrieval models. In the vector space model, a centroid vector is often constructed based on the feedback documents, and a new query vector is then formed by moving the original query vector closer to this feedback centroid vector [11]. In the classic probabilistic model, the feedback documents are naturally treated as examples of relevant documents to improve the estimation of model parameters [10], whereas in the recently proposed language modeling approaches, it can be implemented through estimating a query language model [5, 18] or relevance model [7] through exploiting a set of feedback documents.

Although the idea of pseudo feedback may be implemented differently, a common observation is that it is generally effective in improving retrieval accuracy, which means that, on average, it tends to improve retrieval accuracy, especially the recall. However, it is also a common observation across all retrieval models that pseudo feedback is not completely reliable; this is not really surprising given that many top ranked documents may actually be non-relevant and can be misleading. Indeed, the performance of existing pseudo feedback methods is often affected significantly by some parameters, such as the number of feedback documents to use and the relative weight of original query terms. For example, in a recent study by researchers at the 2003 Reliable Information Access (RIA) Workshop [2, 9], it has been found that nine retrieval systems, representing quite different retrieval models, have all shown tradeoffs in the choices of optimal numbers of documents for pseudo-relevance feedback; usually some amount of documents for feedback can be helpful but too many of them would cause negative influence. They also show that the optimal number of documents used for feedback varies from system to system, and no explicit relationship has been found between either the query length or the number of relevant documents and the optimal number of documents for feedback. These observations clearly suggest that the performance of many existing pseudo feedback methods is sensitive to parameters such as the number of documents for feedback, and it is quite challenging to choose the optimal number of documents for feedback.

There has been some previous work on improving the robustness of pseudo feedback, mostly through heuristics approaches to optimize feedback parameters in a query-specific way, to detect cases when pseudo feedback should not be applied, or to select only a sample of documents from the top ranked documents for pseudo feedback [17, 13]. Unfortunately, these studies are mostly unsuccessful. (A comprehensive discussion of this previous work can be found in [13].) The fundamental reason why the study of robust

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

pseudo feedback has so far been unsuccessful is due to the fact that some or even many top-ranked documents are indeed non-relevant. However, there are also two other common reasons why most existing pseudo feedback methods are not very robust: (1) The combination of the original query and the feedback documents has been "loose". For example, a common strategy is to extract some presumably useful information from the feedback documents and then combine it with the original query representation through interpolation. This strategy is adopted in the Rocchio feedback method [12] and the mixture language model method [18]. One deficiency of such a strategy is that the extraction of information from feedback documents is not guided by the original query terms; as a result, the extracted information can be easily distracted by any dominant distracting theme in the feedback documents. (2) The non-relevant information contained in the feedback documents is not carefully modeled. The feedback documents would inevitably contain many non-relevant documents, but many existing methods do not attempt to discriminate documents in the feedback set or do not model the differences of documents adequately. As a result, the performance tends to be very sensitive to the number of documents used for feedback.

The recent development in language models for information retrieval has been able to address both issues to certain extent. For example, the relevance model approach proposed in [7] and the Markov chain approach proposed in [5] combine the original query with the feedback documents in a principled way. The cluster-based query expansion [4] further uses document cluster information. Mixture models have been explored in [18, 3, 14]. However, even in these more principled models, there are still parameters to tune. For example, in the relevance model [7], interpolation with a collection language model is needed, which introduces an interpolation parameter that needs to be set manually. Similarly, in the Markov chain approach, the probability of stopping "browsing" is a parameter that needs to be set manually. In the mixture model proposed in [18], a background noise parameter and an interpolation parameter are introduced, and both need to be set manually. A problem with such manually set parameters is that we have no guidance on how to set them and the optimal setting tends to vary significantly according to the query and collection. Thus even in the language modeling framework, it remains a challenge to develop robust feedback models – models that either have no parameter to tune, or are relatively insensitive to parameter settings.

In this paper, we show that it is possible to develop a robust pseudo feedback method with no parameter to tune through using appropriate mixture language models and carefully regularizing the estimation of language model parameters. Our work is based on the two-component mixture model proposed in [18], which has been shown to perform quite well in several different tasks as a pseudo feedback method. The basic idea of this model is to first fit a two-component (topic and background) mixture model to the feedback documents and then interpolated the estimated component topic model with the original query model to obtain an improved query language model. This model was later extended in [14] to better integrate the original query model with the feedback documents and to allow each feedback document to potentially contribute differently to the estimated feedback topic language model. The extended model is shown to be relatively more robust than the original model, but the model is still quite sensitive to the number of documents used for feedback [14]. Moreover, due to the use of several priors, this new model has many prior parameters that need to be set manually.

We believe that the key to improve the robustness of such mixture models lies in careful regularization of the parameter estimation process and eliminating as many parameters as possible; ideally, a feedback query model should not rely on any manually assigned parameter, and all model parameters would be learned from the original query model and the feedback documents.

We propose to make two modifications to the two-component mixture model proposed in [18] so as to eliminate the two parameters that would otherwise need to be set manually: (1) Introduce a document-specific mixing coefficient to model potentially different amount of relevance information in each feedback document; the coefficient will be automatically set through statistical estimation. (2) Use the original query model as a prior on the feedback language model to be estimated, which not only reparameterizes the interpolation with a more meaningful parameter (i.e., confidence on the original query model), but also allows us to regularize the estimation of the feedback language model so that useful information from the feedback documents would be gradually incorporated into the original query language model. The confidence parameter is set through stopping the iterative estimation process at the "right" time to balance the original query model and the amount of relevance information extracted from the feedback documents.

On the surface, these extensions appear to be quite similar to what has been done in our previous work [14]. However, there are two key differences: (1) The purpose of using the original query model as a prior is different; here our purpose is mainly to exploit the query model to guide us in distinguishing documents that potentially have more relevance information than those that do not so that the estimated language model would be more influenced by the documents that are more likely to be relevant; indeed, we will gradually reduce the confidence value on the prior to allow more and more information from feedback documents to be incorporated into the query language model. (2) Unlike in [14], the mixing coefficient of the two models does not take a prior and will be estimated solely based on the data.

We further propose a regularized EM algorithm for estimating the parameters, which includes the following two regularization strategies: (1) gradually discounting the prior confidence; and (2) stopping the algorithm as we have sufficient amount of relevance information extracted from feedback documents. Our main idea is to use the original query model to gradually "attract" relevant terms from the feedback documents in a controlled manner. At the same time, the original query model is also naturally mixed with the new model learned from the feedback documents as a prior.

A major advantage of this new model over all existing feedback methods is that it has no parameter to tune. We evaluated this new model and the new estimation method on two representative data sets. The results show that the new method is significantly more robust than a state-of-the-art baseline language modeling approach for feedback with comparable or better accuracy.

In the rest of the paper, we first present the new mixture model in Section 2. We then present a regularized EM algorithm in Section 3. In Section 4, we discuss the experiment results. Finally, we conclude in Section 5.

2. A QUERY-REGULARIZED MIXTURE MODEL FOR PSEUDO FEEDBACK

In this section, we extend previous work [18, 14] and develop a query-regularized mixture model for pseudo feedback.

2.1 The KL-divergence retrieval model

Our basic retrieval model is the Kullback-Leibler (KL) divergence retrieval model [5, 19, 6], which assumes that both a query and a document are generated from a unigram language model. To

score a document D w.r.t. a query Q, we would first estimate the query language model θ_Q and the document language model θ_D, and then compute the KL-divergence $D(\theta_Q||\theta_D)$ as follows:

$$D(\theta_Q||\theta_D) = \sum_{w \in V} p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)}$$

where V is the set of all the words in our vocabulary. For the sake of efficiency, we often truncate the query language model θ_Q to prune a large number of small probability words that would contribute to scoring insignificantly anyway.

Clearly, the two main tasks are to estimate the query language model θ_Q and the document language model θ_D. The document model θ_D is generally estimated with some smoothing method, resulting in some form of interpolation of the relative frequency of a word in the document and the probability of the word given by some background language model [19]. For example, using one of the most effective smoothing methods called Dirichlet prior smoothing method, we would have

$$p(w|\theta_D) = \frac{c(w,D) + \mu p(w|\theta_B)}{|D| + \mu}$$

where $c(w,D)$ is the count of word w in D, $p(w|\theta_B)$ is a collection background language model, and μ is a smoothing parameter often set empirically. The background language model θ_B is estimated using the whole collection of documents C [19], i.e., $p(w|\theta_B) = \frac{\sum_{D \in C} c(w,D)}{\sum_{D \in C} |D|}$.

The query model intuitively captures what the user is interested in, thus would affect retrieval accuracy significantly. Without feedback, θ_Q is often estimated as $p(w|\theta_Q) = p(w|Q) = \frac{c(w,Q)}{|Q|}$, where $c(w,Q)$ is the count of word w in the query Q, and $|Q|$ is the total number of words in the query. Such a model, however, is not very discriminative because a query is typically extremely short. Several different methods have been proposed to improve the estimation of θ_Q by exploiting documents, especially those documents that are used for pseudo-relevance feedback (i.e., the top-ranked documents from an initial retrieval run) [5, 18, 7]. In [18], it is proposed that pseudo-relevance feedback can be implemented in the KL-divergence retrieval model as updating the query model based on the feedback documents. Specifically, we can define a two-component mixture model (i.e., a fixed background language model θ_B set using the whole collection and an unknown topic model to be estimated) and assume that the pseudo feedback documents are generated using such a mixture model. Formally, let θ_T be the unknown topic language model and $F \subset C$ a set of pseudo feedback documents. The log-likelihood function of the mixture model is

$$\log p(F|\theta_T) = \sum_{D \in F} \sum_{w \in V} c(w,D) \log((1-\alpha)p(w|\theta_B) + \alpha p(w|\theta_T))$$

where α is a parameter indicating how likely we would use the topic model (as opposed to the background model) to generate a word in a document in F. In [18], α is set to a constant, and the Expectation-Maximization (EM) algorithm is used to estimate θ_T, which is then interpolated with the empirical query model $p(w|Q)$ to obtain an improved estimate of the query model $p(w|\theta_Q) = (1-\lambda)p(w|Q) + \lambda p(w|\theta_T)$, where λ is another parameter to be manually set.

Although this model has been shown to perform quite well, its performance is reported to be sensitive to the setting of both α and λ [18]. This motivates us to develop the following query-regularized mixture model as an extension of this basic model; the new model can eliminate the need for manually setting these two parameters. Specifically, we will estimate a potentially different α for each document, rather than manually set it to a fixed constant, and we will incorporate the empirical query model $p(w|Q)$ into the mixture model as a prior. We will also make our target θ_Q a component in the mixture model so that we can avoid "post estimation interpolation". We now present this new mixture model.

2.2 Query-regularized mixture model

Intuitively, all pseudo feedback methods would try to learn some useful information from the feedback documents. The two component mixture model with a background model proposed in [18] would allow us to learn a unigram component language model (i.e., θ_T) from the feedback documents by factoring out words with high probabilities according to the background model.

One deficiency of this simple mixture model is that the mixing coefficient α is fixed across all the documents even though some feedback documents presumably have more noise than others. To model different amount of relevance in different documents, we should allow each document to have a potentially different α_D. Naturally, we would expect a relevant document to have a larger α_D than a non-relevant one. Formally, suppose D is a document, we have

$$\log p(D|\theta_T, \alpha_D) = \sum_{w \in V} c(w,D) \log(\alpha_D p(w|\theta_T) + (1-\alpha_D)p(w|\theta_B))$$

where α_D is a document-specific mixing weight parameter. The log-likelihood for all the feedback documents in F is thus

$$\log p(F|\Lambda) = \sum_{D \in F} \sum_{w \in V} c(w,D) \log(\alpha_D p(w|\theta_T) + (1-\alpha_D)p(w|\theta_B))$$

where $\Lambda = \{\theta_T, \{\alpha_D\}_{D \in F}\}$ are all the parameters to estimate.

Another deficiency of the simple mixture model is that it does not involve the original query in any way. As a result, the estimation of θ_T is completely independent of the original query, and we must interpolate the learned θ_T with the original query model $p(w|Q)$ in an ad hoc way. In order to integrate the original query with the learned language model from feedback documents, we use the original (empirical) query model $p(w|Q)$ to define a conjugate Dirichlet prior for θ_T, $Dir(\{1 + \mu p(w|Q)\}_{w \in V})$. That is,

$$p(\theta_T) \propto \prod_{w \in V} p(w|\theta_T)^{\mu p(w|Q)}$$

In effect, this is to force the estimated θ_T to be as close to $p(w|Q)$ as possible. Here μ is a parameter indicating our confidence on the original query model prior. Since the prior is conjugate, μ can be interpreted as "equivalent sample size", i.e., the influence of the prior would be equivalent to adding $\mu p(w|Q)$ pseudo counts for word w when estimating the topic model θ_T. Intuitively, the larger μ is, the more constrained θ_T is by $p(w|Q)$. We will discuss later how we will use μ to allow the original query model $p(w|Q)$ to dynamically regularize the estimation of θ_T.

We do not put any informative prior on α_D's. Thus our prior on all the parameters is

$$p(\Lambda) \propto p(\theta_T) \propto \prod_{w \in V} p(w|\theta_T)^{\mu p(w|Q)}$$

With this prior, we may use Bayesian estimation to maximize the posterior probability of parameters, as opposed to maximizing the likelihood function of the parameters as in [18]. Specifically, we can use the following Maximum A Posterior (MAP) estimator:

$$\hat{\Lambda} = \arg\max_{\Lambda} p(F|\Lambda)p(\Lambda) \quad (1)$$

Since we use a conjugate prior, the MAP estimate can also be computed by modifying the M-step in the EM algorithm to incorporate

extra counts as defined by the prior (see [8] for the derivation). The updating formulas are as follows:
E-step:

$$p(Z_{w,D}=1) = \frac{\alpha_D^{(n)} P^{(n)}(w|\theta_T)}{\alpha_D^{(n)} P^{(n)}(w|\theta_T) + (1-\alpha_D^{(n)}) P(w|\Theta_B)}$$

M-step:

$$\alpha_D^{(n+1)} = \frac{\sum_{w\in V} p(Z_{w,D}=1) c(w,D)}{\sum_{w\in V} c(w,D)}$$

$$P^{(n+1)}(w|\theta_T) = \frac{\mu P(w|Q) + \sum_{D\in F} c(w,D) p(Z_{w,D}=1)}{\mu + \sum_{w'\in V}\sum_{D\in F} c(w',D) p(Z_{w',D}=1)}$$

The hidden variable $Z_{w,D}$ indicates whether term w in document D has been generated using θ_T as opposed to θ_B. Since we have incorporated the original query model $p(w|Q)$ as a prior, the estimated topic model θ_T could be taken *directly* as our updated query model based on feedback documents. That is, we no longer need to interpolate θ_T with the original query model; instead, we will use θ_T to replace θ_Q in computing the KL-divergence value.

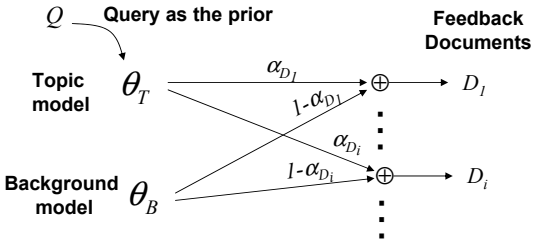


Figure 1: Mixture model for pseudo feedback.

The mixture model is illustrated in Figure 1. On the surface, this model is very similar to the one proposed in [14], where each document is also allowed to have a different mixing coefficient. However, there are a few important differences between them: (1) The model in [14] puts a prior on α_D, which introduces several parameters that need to be manually set, whereas we attempt to estimate α_D without any prior. This not only eliminates the need for manual parameter setting, but also gives our model more freedom to discriminate feedback documents in terms of the amount of relevance information, which is crucial to achieve robustness w.r.t. the number of documents for feedback. (2) The estimation method used in [14] performs a standard EM MAP estimation, while we use a regularized EM algorithm (to be described below) that allows us to control the parameter estimation process and gradually let the prior query model attract relevant terms from the feedback documents.

3. REGULARIZED EM PARAMETER ESTIMATION

The mixture model presented above gives us a way to extract a query language model from the feedback documents that is close to our original query language model. However, if we apply the standard EM algorithm directly to estimate the query language model, we run into two problems:

Setting of μ: We will have to set the prior confidence value μ manually. μ controls the relative weight we put on the original query language model, so in this sense, it plays a role similar to the interpolation parameter in the previous work [18]. However, since the relative weight on the original query language model depends on both μ and the accumulated counts from the feedback documents during the EM algorithm, it would be non-optimal to set μ to some predefined constant. Intuitively, it should be set based on the accumulated relevance counts from the feedback documents.

Accuracy of α_D: Although we allow each document to potentially contribute different amount of relevance information to the estimated query language model through a document-specific α_D, how accurately α_D reflects the amount of relevance information in a document highly depends on the accuracy of the tentatively estimated query language model θ_T at each iteration. Once θ_T drifts away from the original query language model, α_D would also be inaccurate, and thus after more iterations, the estimated query language model will be quite biased toward the popular topic in the feedback documents. Thus our key idea to achieve robust feedback is to ensure that the tentative query model at each iteration of EM is reasonably accurate, i.e., the influence of the original query language model as a prior should be kept at a high level all the time, which can be achieved by setting μ to a large value. Unfortunately, a large μ would prevent θ_T from attracting many useful query-related words from the feedback documents, because a large μ would make θ_T more rigid, thus creating an upper bound on α_D and also an upper bound on the amount of words θ_T can attract from the feedback documents.

To solve these two problems, we propose to start with a very high value for μ and gradually lower μ in each EM iteration. The idea is to gradually relax our constraint on θ_T so that it can slowly assimilate useful words from feedback documents. Because of such regularization, we may expect the tentatively estimated θ_T at any time to be relatively reliable.

As we regularize EM in this way, we would force α_D's to all start with very small values since θ_T is initially almost the same as the original query model and cannot explain many words in the documents. As we lower μ, however, α_D's would become larger and larger, causing the expected relevance count in the second equation of the M-step (i.e., $r^{(n)} = \sum_{w'\in V}\sum_{D\in F} c(w',D) p(Z_{w',D}=1)$) to increase as well, which allows θ_T to become more and more adapted toward the feedback documents. Since at each iteration, μ is decreasing while the relevance count $r^{(n)}$ is increasing, they eventually will "meet" (i.e., $r^{(n)} \geq \mu$). At this time, we can stop the EM algorithm since we have now accumulated about the same amount of relevance information as the equivalent sample size of our prior based on the original query model. In other words, this is the time when we have roughly an equal weight on the original query and the new information extracted from the feedback documents and it is a relatively "safe" stopping point.

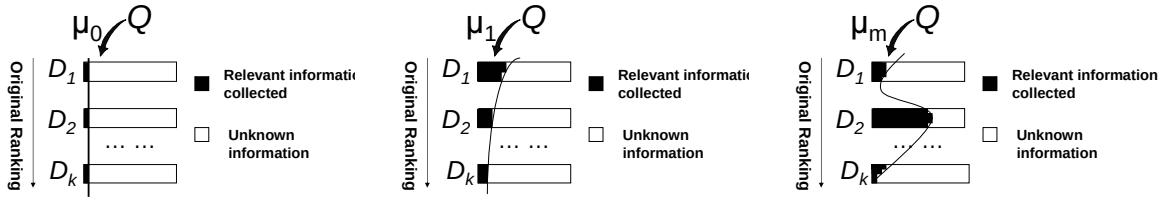
Actually we may consider letting $r^{(n)}$ grow even more to achieve more aggressive feedback. However, the additional words attracted would no longer be so reliable because our constraint of prior on the query model is no longer so strong as when μ is much larger than r, so the algorithm would likely become less robust. Indeed, our experiments show that when we use few top-ranked documents for pseudo-relevance feedback, we can often further improve performance by allowing $r^{(n)}$ to grow more aggressively, but when we use a large number of documents for feedback, more conservative growth of $r^{(n)}$ is often better.

Note that although stopping the algorithm when $r^{(n)} \geq \mu$ is in effect achieving a roughly equal weight interpolation between the original query model and the feedback model, the learned feedback model in our method is expected to be more robust than one learned from the EM algorithm where θ_T is not regularized such as in [18]. This is confirmed in our experiments. Our regularized EM also helps address the multiple local maxima problem by regularizing the path with the original query model and target at a specific local

Table 1: Comparison of RMM and KL, SMM-fixed, and SMM-opt (MAP).

	#docs	KL	SMM-opt	SMM-fixed	RMM	Impr. (over KL)	Impr. (over SMM-fixed)
DOE	10	0.1803	0.2004	0.1907	0.1975	9.53% *	3.565% *
	50	0.1803	0.2142	0.2142	0.1977	9.65% *	-7.70%
	100	0.1803	0.1969	0.1938	0.1960	8.70% *	1.135%
	150	0.1803	0.1903	0.1903	0.1962	8.81% *	3.100%
	200	0.1803	0.1861	0.1858	0.1945	7.87% *	4.682% *
	300	0.1803	0.1857	0.1778	0.1930	7.04% *	8.548% *
TREC678	10	0.2242	0.2392	0.2392	0.2427	8.25% *	1.46%
	50	0.2242	0.2313	0.2313	0.2391	6.64% *	0.79%
	100	0.2242	0.2301	0.2185	0.2371	5.75% *	8.51% *
	150	0.2242	0.2304	0.2181	0.2352	4.90% *	7.84% *
	200	0.2242	0.2269	0.2169	0.2321	3.52%	7.00% *
	300	0.2242	0.2254	0.2136	0.2293	2.27%	7.35% *

* The improvement is statistically significant at the level of 0.05 according to the Wilcoxon signed rank test.



(a) step 0 (b) step 1 (c) step m

Figure 2: Gradually collecting relevance information.

maximum that is most consistent with our original query model.

In sum, our main idea for achieving robust pseudo feedback is to start with setting θ_T to the original query model, and then iteratively discriminate documents based on how well they match the current θ_T and expand θ_T with relevant words from the feedback documents accordingly. The process stops as we have accumulated enough relevance information from the feedback documents that balances well with the original query model. We illustrate this regularized estimation process in Figure 2. In the early stage of training, the only reliable information we have is the query prior. Thus, we first assign a large prior confidence μ. This confidence is large enough to force θ_T to follow the original query without incorporating much relevance information from feedback documents (Figure 2 (a)). As this process continues, we would gradually relax the control of prior information by discounting μ. During each iteration, μ is reduced a little bit so that some additional relevance information can be added in. We use a discounting factor $\delta \in (0, 1)$ to control the discounting rate: $\mu^{n+1} = \sigma\mu^n$. Our experiments show that the performance is insensitive when the initial μ (i.e., μ^0) is sufficiently large and δ is close to 1 (i.e., when we slowly discount a large query prior confidence value).

4. EXPERIMENT RESULTS

We tested our new method on two standard TREC data sets [15]: DOE (Department of Energy) data, and TREC678 (the ad hoc data used in TREC6, TREC7, and TREC8). DOE has 226,087 documents, and each document is about 117 terms on average. We used 35 queries that have a descent number of relevant documents, and the total number of relevant documents for these queries is 2,047. TREC678 has 528,155 documents with an average document length of about 477 terms. We used TREC 301-450 topics, which have a total number of 13,692 relevant documents. We chose TREC678 for the reason that this data has been used in the RIA Workshop, in which the robustness of pseudo-relevance feedback was studied [2, 9]. For both data sets, we used only the title fields of the topics to simulate the kind of extremely short keyword queries that users often use.

We implemented our new feedback model on top of the Lemur toolkit[1], and used Lemur for all our experiments. Following the TREC standard, we retrieve 1,000 documents for each query, and use mean average precision (*MAP*) as the primary performance measure. In all the experiments, we first use the basic KL-divergence method without feedback to retrieve a ranked list of documents for each query as our feedback documents. Dirichlet Prior smoothing is used and the smoothing parameter is set to 2000 as recommended in [19]. We then use either our regularized mixture model (RMM) or the simple mixture model (SMM) to perform pseudo-relevance feedback with a certain number of top-ranked documents from the initial retrieval results. We truncate the updated query language model to keep 100 highest probability terms and use the KL-divergence method again to perform a second round retrieval. We compare RMM with SMM on both data sets to study the effectiveness of the new model.

4.1 Robustness Analysis

We first compare RMM with SMM in terms of their robustness w.r.t. the number of documents used for feedback. We tested RMM and SMM with six different numbers of feedback documents: 10, 50, 100, 150, 200, 300. The MAP results are shown in Table 1, where we compare RMM with (1) the baseline no-feedback run (KL), (2) SMM with fixed parameters tuned to optimize performance for 50 feedback documents (SMM-fixed), and (3) SMM tuned for each different number of feedback documents (SMM-opt).

As discussed in Section 2, SMM has two major parameters to tune, one for the background noise in the mixture model and one for the interpolation weight. Without relevance information about

[1]Available at http://www.lemurproject.org/

the test queries, these parameters generally need to be set based on some training queries. In our experiments, we give SMM a slightly unfair advantage by tuning its two parameters to optimize its performance on *test* queries for 50 feedback documents and then fix the parameter values in all the experiments with other numbers of feedback documents. This strong baseline run is labeled as "SMM-fixed". We also include an optimal run of SMM (labeled as "SMM-opt"), in which the two parameters are tuned in each experiment on the test queries. SMM-opt thus represents the upper bound performance of SMM on the test queries.

Strictly speaking, RMM also has two insensitive parameters to set: the initial prior confidence (μ^0) and the discounting factor (δ). We set μ^0 and δ to 30,000 and 0.9, respectively. As will be shown later, the feedback performance is not sensitive to these two parameters as long as we choose a large μ^0 and a δ that is close to 1.0.

From Table 1, we can make the following observations:

We notice that RMM is always better than the no-feedback baseline KL, and the improvement is statistically significant in most cases, indicating that as a pseudo feedback method, RMM is effective and robust w.r.t. using different number of documents for feedback. In contrast, SMM-fixed is noticeably less robust; indeed, although it improves over the no-feedback baseline KL when using a relatively small number of documents for feedback, it actually does not perform as well as the KL baseline when a large number of documents (e.g., 300) are used for feedback. Even SMM-opt, with all the parameters optimized specifically for the number of documents used for feedback, can barely improve over KL. The sensitivity of SMM to the number of feedback documents also seems to be more pronounced on TREC678 than on DOE. One possible explanation is that as we use more feedback documents, the noise becomes more harmful in the case of TREC678 because DOE is a relatively small, homogeneous data set whereas TREC678 is much larger and more heterogeneous.

Compared with SMM-fixed, RMM is significantly more robust as shown in the last column of the table, where we see that the improvement of RMM over SMM-fixed is often amplified as we use more and more feedback documents, clearly indicating the superior robustness of RMM. There is only one case (DOE, 50 docs) when RMM is worse than SMM-fixed, but in this case, SMM-fixed has been tuned on the *test* queries. In reality, it is unlikely that we can perform such tuning, thus SMM will most likely perform worse than SMM-fixed in real applications. Considering the fact that RMM has no parameter to tune, the improvement of RMM over SMM-fixed is quite impressive.

However, the performance of RMM is still affected by the number of feedback documents and the performance seems to decrease monotonically as we use more and more feedback documents, indicating that there is still room for further improvement of our method in terms of robustness.

We further compare RMM with SMM-fixed in terms of precisions at different levels of recall in Table 2. We see that once again, RMM outperforms SMM-fixed significantly in most cases.

Table 2: Precision at different recall levels.

	Recall	SMM-fixed	RMM	Improvement(%)
DOE (#docs =300)	0.0	0.5225	0.5844	11.84% *
	0.1	0.3518	0.4071	15.71% *
	0.2	0.3071	0.3365	9.573% *
	0.3	0.2047	0.2217	8.304% *
	0.4	0.1825	0.1952	6.958% *
	0.5	0.1606	0.1736	8.094% *
	0.6	0.1386	0.1486	7.215% *
	0.7	0.1165	0.1192	2.317%
	0.8	0.1033	0.1012	-2.03%
	0.9	0.0661	0.0617	-6.65%
	1.0	0.0276	0.0278	0.724%
TREC678 (#docs =300)	0.0	0.2263	0.2369	4.6840% *
	0.1	0.6041	0.6537	8.2105% *
	0.2	0.4305	0.4696	9.0824% *
	0.3	0.3598	0.3870	7.5597% *
	0.4	0.3148	0.3263	3.6531% *
	0.5	0.2662	0.2726	2.4042% *
	0.6	0.2316	0.2443	5.48% *
	0.7	0.1744	0.1766	1.2614%
	0.8	0.1394	0.1381	-0.932%
	0.9	0.0948	0.0973	2.6371%
	1.0	0.0581	0.0649	11.704% *

* The improvement is statistically significant at the level of 0.05 according to the Wilcoxon signed rank test.

4.2 Estimation details

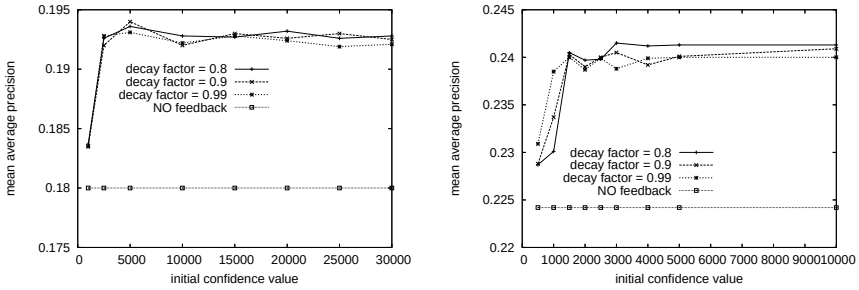
4.2.1 Prior confidence discounting

A main idea of RMM is to start with a high confidence value for the query model prior (μ^0) and gradually discount μ by a factor of δ in each iteration of the EM algorithm. (See Section 3.) Strictly speaking, these are parameters in our method, thus one question is to what extent the performance is affected by the setting of μ^0 and δ. In Figure 3, we plot the MAP values for different values of μ^0 and δ for RMM on both DOE (#docs=300) and TREC678 (#docs=10). We see that the performance is insensitive to the setting of these parameters as long as the initial confidence value μ^0 is set to a sufficiently large number and the discounting factor δ is set to a value close to 1.0., i.e., we start with a sufficiently high confidence value and discount the confidence *slowly*, which makes sense intuitively and is what we would expect.

4.2.2 Convergence of EM

One may notice that the proposed estimation process stops immediately after the collected relevance information reaches the equivalent sample size of the query prior. Intuitively, with slow discounting of the prior confidence, this process would result in an improved query language model through gradual incorporation of relevance information from the feedback documents into the original query model. Our experiment results also show that the method is effective and robust for pseudo-relevance feedback.

However, since we did not let the EM algorithm converge, it is unclear how to interpret the estimated query model θ_T from the viewpoint of MAP estimation. We thus run some additional experiments, in which we further continue the EM algorithm until it converges, but without further lowering the query confidence (labeled as RMM-conv). The results are shown in Table 3. Comparing RMM-conv with RMM, we see that they are mostly similar, suggesting that the estimated θ_T with early stopping (RMM) may actually be quite close to the local maximum which RMM-conv obtains. However, RMM-conv seems to be slightly more sensitive to the number of feedback documents than RMM, and it tends to outperform RMM slightly when fewer documents are used for feedback, but would lose to RMM slightly as more documents are used. One possible explanation is that as the EM algorithm continues, the collected amount of relevance information from the feedback documents would likely increase, causing more aggressive feedback, thus higher sensitivity to the number of feedback doc-



(a) DOE (b) TREC678

Figure 3: Performance sensitivity to prior discounting.

Table 3: EM convergence and "inside-estimation" interpolation.

Data	#docs	KL	SMM-0.5	RMM	RMM-conv
DOE	10	0.1803	0.2000	0.1975	0.1982
	50	0.1803	0.2076	0.1977	0.2019
	100	0.1803	0.1948	0.1960	0.2004
	150	0.1803	0.1873	0.1962	0.1958
	200	0.1803	0.1855	0.1945	0.1936
	300	0.1803	0.1847	0.1930	0.1921
TREC678	10	0.2242	0.2316	0.2427	0.2462
	50	0.2242	0.2283	0.2391	0.2399
	100	0.2242	0.2277	0.2371	0.2368
	150	0.2242	0.2262	0.2352	0.2345
	200	0.2242	0.2260	0.2321	0.2323
	300	0.2242	0.2254	0.2293	0.2279

uments. However, the difference is so small that more experiments are clearly needed to further analyze this difference.

4.2.3 Query prior vs. interpolation

One difference between RMM and SMM is that RMM incorporate the original query model as a prior to achieve an "inside-estimation" interpolation effect, while in SMM, the interpolation is done after the estimation process is finished. Presumably, the "inside-estimation" interpolation is better since the estimation process can be guided by the original query model. To see whether such inside-estimation interpolation has any empirical advantage, we also show in Table 3 an optimal run of SMM with a fixed value (0.5) for the interpolation parameter (labeled as SMM-0.5), in which we optimize the background noise parameter on the test queries. This run is comparable with RMM because when RMM stops, the learned query model would correspond roughly to a $(0.5, 0.5)$ interpolation of relevance information learned from the feedback documents with the original query model. Comparing RMM with SMM-0.5, we see that the inside-estimation interpolation is indeed advantageous probably due to the benefit of guiding the estimation of θ_T with the query model prior.

4.2.4 Upper bound analysis

Since our method stops EM iterations when a prior confidence μ is equal to the relevance count $r^{(n)}$, an interesting question is whether stopping exactly in the middle is optimal. Indeed, we can introduce an additional parameter η to accommodate more aggressive feedback *i.e.* EM would stop when $\mu \times \eta \leq r^{(n)}$. With this extension of RMM, we can adjust η to obtain the best performance for different number of feedback documents, which in some sense reflects an upper bound of the performance for RMM. The results are shown in Table 4. It is interesting to see that the optimal η is correlated with the number of feedback documents. The optimal η tends to be higher (trusting the feedback information more) when using fewer top-ranked documents for feedback, which intuitively makes sense as there would be more noise when we use more documents for feedback. Overall, setting $\eta = 1.0$ as in RMM is mostly close to optimal. This analysis, however, does suggest an interesting possibility of automatically setting the parameter η based on the number of feedback documents, which may further improve performance of RMM.

Table 4: Performance upper bound of RMM

#docs	DOE		TREC678	
	RMM	RMM-opt (η)	RMM	RMM-opt (η)
10	0.1975	0.2015 (5.0)	0.2427	0.2476 (2.0)
50	0.1977	0.2032 (3.0)	0.2391	0.2398 (1.3)
100	0.1960	0.1986 (1.5)	0.2371	0.2382 (0.9)
150	0.1962	0.1962 (1.0)	0.2352	0.2352 (0.9)
200	0.1945	0.1952 (0.9)	0.2321	0.2331 (0.9)
300	0.1930	0.1947 (1.3)	0.2293	0.2319 (0.8)

4.3 Individual query analysis

To better understand the difference between our method and the simple mixture model, we further studied in detail two extreme queries (i.e., query 447 and query 312). Query 447 ("stirling engine") is an example query on which our model is significantly better than the simple mixture model, while Query 312 ("hydroponics") is one on which our model is significantly outperformed by the simple mixture model. In Table 5, we show the learned query models for both SMM-fixed and RMM.

SMM-final is the final model from SMM after interpolation with the original query model with an optimal mixing coefficient (0.1). Since the optimal mixing coefficient has a small value, it is not surprising that the original query words are quite dominant in SMM-final. RMM-final is the estimated query model using RMM. As expected, the coefficient of the implicit interpolation is around 0.5. We also show the initial query model for RMM (RMM-init), which is mostly the original query model. Comparing RMM-final with SMM-final for Query 447, we see that RMM-final has included more discriminative words in the model, while SMM-final tends to be dominated by some general terms, which may explain why RMM outperformed SMM. In the case of Query 312, however, it is unclear why RMM-final has performed worse than SMM-final. It seems that neither has picked up many good terms for this query; instead they both have some distracting terms. Since RMM has assigned higher weights to the expanded terms than SMM, these

Table 5: Sample query models for two representative queries

Query 447			Query 312		
RMM-init	**RMM-final**	**SMM-final**	**RMM-init**	**RMM-final**	**SMM-final**
stirl 0.4997	stirl (0.2902)	stirl (0.44973)	hydropon (0.9999)	hydropon (0.5331)	hydropon (0.8999)
engin 0.4993	engin (0.2738)	engin (0.44937)	khouraki ($\sim$ 0.0)	tomato (0.0174)	and (0.00201)
brackenre ($\sim$ 0.0)	cfchyph (0.0232)	the (0.00145)	fortevil ($\sim$ 0.0)	pepper (0.0141)	in (0.00087)
msemaji ($\sim$ 0.0)	substitut (0.0122)	p (0.00143)	moqti ($\sim$ 0.0)	marijuana (0.0139)	tomato (0.00080)
takiab ($\sim$ 0.0)	rhyph (0.0117)	a (0.00141)	oakcliff ($\sim$ 0.0)	meat (0.0126)	a (0.00075)
cabletransmit ($\sim$ 0.0)	accept (0.0115)	for (0.00134)	altayebat ($\sim$ 0.0)	cook (0.0120)	or (0.00073)
evaporativefre ($\sim$ 0.0)	hfchypha (0.0075)	and (0.00108)	muhammara ($\sim$ 0.0)	teaspoon (0.0091)	stirl (0.00072)
barnswood ($\sim$ 0.0)	hcfchyph (0.0068)	i (0.001079)	aleppostyl ($\sim$ 0.0)	indoor (0.0087)	marijuana (0.00065)
boghall ($\sim$ 0.0)	retrofit (0.0067)	in (0.0009)	schamadan ($\sim$ 0.0)	onion (0.0083)	pepper (0.00064)
lekand ($\sim$ 0.0)	refriger (0.0062)	of (0.00087)	alubalu ($\sim$ 0.0)	altayebat (0.0063)	drug (0.00063)
hydroson ($\sim$ 0.0)	blank (0.0061)	develop (0.00083)	chiller ($\sim$ 0.0)	salt (0.0093)	cfchyph (0.00061)
hallstavik ($\sim$ 0.0)	rampd (0.0054)	pound (0.0008)	centrifug ($\sim$ 0.0)	teaspoon (0.0092)	meat (0.00058)
rockhammar ($\sim$ 0.0)	chiller (0.0045)	amp (0.0007)	sdgamp ($\sim$ 0.0)	indoor (0.009)	cup (0.00056)
technologyse ($\sim$ 0.0)	centrifug (0.0042)	energi (0.00073)	killea ($\sim$ 0.0)	cup (0.0085)	cook (0.00055)
hzat ($\sim$ 0.0)	sdgamp (0.0039)	technologi (0.00064)	kw ($\sim$ 0.0)	onion (0.0084)	serv (0.00051)
rimfir ($\sim$ 0.0)	killea (0.0039)	mr (0.00062)	rhypha ($\sim$ 0.0)	hfchypha (0.0075)	said (0.00050)
kgmmsup ($\sim$ 0.0)	kw (0.0037)	cfchyph (0.00061)	hcfc ($\sim$ 0.0)	hcfchyph (0.0068)	us (0.00049)
plantcoal ($\sim$ 0.0)	rhypha (0.0037)	wa (0.00058)	kronor ($\sim$ 0.0)	retrofit (0.0067)	beef (0.00046)
waterlithium ($\sim$ 0.0)	hcfc (0.0035)	which (0.00049)	reclam ($\sim$ 0.0)	altayebat (0.0063)	arrest (0.00046)
metabolist ($\sim$ 0.0)	blend (0.0035)	power (0.00046)	rhyphb ($\sim$ 0.0)	oliv (0.0062)	grow (0.00045)

distracted terms may hurt the performance more than in the case of SMM.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we present a robust method for pseudo feedback based on statistical language models. The new method uses the original query model as a prior on the feedback language model to be estimated, which not only reparameterizes the interpolation with a more meaningful parameter, but also allows us to regularize the estimation of the feedback language model so that useful information from the feedback documents would be gradually incorporated into the original query language model. We further propose a regularized EM algorithm for estimating the parameters, which includes the following two regularization strategies: (1) gradual discounting the prior confidence, and (2) stopping the algorithm based on the amount of relevance information extracted from feedback documents (controlled by a parameter). Experiment results on several representative data sets show that the new method is more robust than a state-of-the-art baseline language modeling approach for feedback with comparable or better retrieval accuracy.

The robustness of the proposed model is partly from using a conservative stopping condition. However, as indicated in the analysis of the upper bound of the proposed model, when the feedback information is reliable, it is desirable to allow more feedback information to be incorporated into the query model. The right amount of feedback information to incorporate is presumably associated with the precision of the feedback documents, which is further related to query difficulty. A very interesting future research direction is to further study how to measure the quality of feedback documents and allow some flexibility in combining the query model prior with feedback information accordingly.

6. ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under award number IIS-0347933. We thank the anonymous SIGIR 06 reviewers for their useful comments.

7. REFERENCES

[1] D. A. Evans and R. G. Lefferts. Design and evaluation of the clarit-trec-2 system. In D. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, 1994.

[2] D. Harman and C. Buckley. The NRRC reliable information access (RIA) workshop. In *Proceedings of ACM SIGIR 2004*, 2004.

[3] D. Hiemstra. Term-specific smoothing for the language modeling approach to information retrieval: the importance of a query term. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35–41, New York, NY, USA, 2002. ACM Press.

[4] O. Kurland, L. Lee, and C. Domshlak. Better than the real thing?: iterative pseudo-query processing using cluster-based language models. In *In proceeding of SIGIR '05*, pages 19–26, 2005.

[5] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'2001*, pages 111–119, Sept 2001.

[6] V. Lavrenko. *A Generative Theory of Relevance*. PhD thesis, Univ. of Massachusetts at Amherst, 2004.

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

[8] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1996.

[9] J. Montgomery, L. Si, J. Callan, and D. A. Evans. Effect of varying number of documents in blind feedback: analysis of the 2003 NRRC RIA workshop "bf_numdocs" experiment suite. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 476–477, New York, NY, USA, 2004. ACM Press.

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

[11] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc., 1971.

[12] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc., 1971.

[13] T. Sakai, T. Manabe, and M. Koyama. Flexible pseudo-relevance feedback via selective sampling. *ACM Transactions on Asian Language Information Processing*, 4(2):111–135, 2005.

[14] T. Tao and C. Zhai. Mixture clustering model for pseudo. In *Proceedings of the 2004 Meeting of the International Federation of Classification Societies*. Spriner, 2003.

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

[16] J. Xu and W. Croft. Query expansion using local and global document analysis. In *Proceedings of the SIGIR 1996*, pages 4–11, 1996.

[17] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18(1):79–112, 2000.

[18] C. Zhai and J. Lafferty. Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410, 2001.

[19] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'2001*, pages 334–342, Sept 2001.