Improving Retrieval Accuracy of Difficult Queries through Generalizing Negative Document Language Models

Maryam Karimzadehgan
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
mkarimz2@illinois.edu

ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
czhai@cs.illinois.edu

ABSTRACT

When a query topic is difficult and the search results are very poor, negative feedback is a very useful method to improve the retrieval accuracy and user experience. One challenge in negative feedback is that negative documents tend to be distracting in different ways, thus as training examples, negative examples are sparse. In this paper, we solve the problem of data sparseness in the language modeling framework. We propose an optimization framework, in which we learn from a few top-ranked non-relevant examples, and search in a large space of all language models to build a more general negative language model. This general negative language model has more power in pruning the non-relevant documents, thus potentially improving the performance for difficult queries. Experiment results on representative TREC collections show that the proposed optimization framework can improve negative feedback performance over the state-of-the-art negative feedback method through generalizing negative language models.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Negative Feedback, Language Models, Difficult topics, Generalizing Language Model, Optimization.

1. INTRODUCTION

When a query is so difficult that a large number of top-ranked documents are non-relevant, a user would have to either reformulate the query or go far down on the ranked list to examine more documents, both may decrease the user satisfaction. As a result, improving the effectiveness of search results for such difficult queries would bring user satisfaction which is the ultimate goal of search engines.

A commonly used strategy to improve search results is through feedback techniques, including relevance feedback (e.g., [29, 30, 31]), pseudo-relevance feedback (e.g., [1, 3, 42]) and implicit feedback [33]. In the case of difficult queries, if we can perform effective negative feedback when a user could not find any relevant document on the first page of the search results, we would be able to improve the ranking of the unseen results in the next few pages. It is clear that in this case of negative relevance feedback, we only have negative (i.e., non-relevant) documents since a query is difficult that none of the top-ranked documents are relevant. When a user is unable to reformulate an effective query (which happens often in informational queries due to insufficient knowledge about the relevant documents), negative feedback can be quite beneficial, and the benefit can be achieved without requiring extra effort from users (e.g., by assuming the skipped documents by a user to be non-relevant).

While relevance feedback has been studied extensively, negative feedback has just attracted attention recently. In [40, 41], the authors studied different methods for negative feedback in both language models and vector space model and concluded that negative feedback for language modeling approaches works better than the vector space model. Negative feedback works by excluding documents that are similar to an example negative document. Previous work [41] has shown MultiNeg strategy is most useful when each individual negative document is considered independently, suggesting that negative documents are distracting in different ways. Thus, as training examples, negative examples are sparse. Intuitively, if we can learn from each single negative example to prune aggressively a lot of non-relevant documents from the top-ranked documents, we should improve the performance more, but in reality there is a risk of over-generalization of a negative example. Thus, an important, yet difficult question is how to appropriately generalize a negative language model; specifically, there are two technical challenges to be solved:

1. What does a general language model mean? How do we formally define generality?

2. Among all the general negative language models of a given negative example, which one should we choose so that we can both maximize its pruning power and avoid over-generalization?

In this paper, we tackle these two challenges in the language modeling framework. To address the first research question, we propose a formal definition of *generality of a*

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

language model to measure if one negative language model is more general than another. For example, if the query is "jaguar" and the user is looking for documents about jaguar animals, a document containing a jaguar car would be a non-relevant example. This document, however, may not mention the word "car" that often, so if we construct a negative document language model based on this document, the high probability words may be words of a particular jaguar car model such as "Alezon" and "Oxford models". While it is safe to use such words to prune non-relevant documents, their pruning power is limited. A more common word like "car" would be able to prune non-relevant documents more effectively. This generalized negative language model is meant to capture this intuition, and it can be expected to be more effective than the original negative document language model in removing other unseen non-relevant documents, thus improving the accuracy of search results.

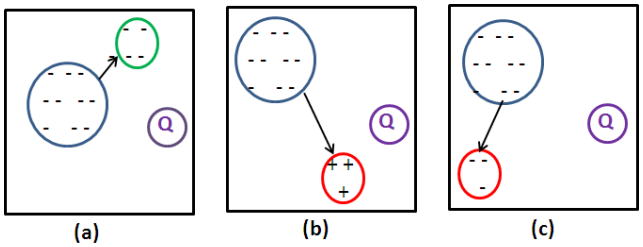

Figure 1: An illustrative example. Only case (a) is desirable.

To address the second research question, we propose an optimization framework where we seek a generalized negative language model that optimizes three criteria: **1)** closeness to the original negative language model, **2)** closeness to the query (if it is far from the query, the pruning power is not very effective), and **3)** a generalization constraint that defines the amount of generalization we would like to achieve. The reason why all these three components are important can be explained in Figure 1, where (a) shows that the general negative language model (i.e., green circle) is safe since it is both close to the original negative language model (thus ensures that the pruned documents to be non-relevant) and reasonably close to the query (thus can make a difference in the top-ranked results through pruning). Figure 1 (b) shows that if we move too far away from the original negative language model and very close to the query, there would be a danger that we would move into the relevant documents zone (red circle). Figure 1 (c) shows another undesirable case where we move too far away from the query and as a result, pruning would only affect the lowly ranked documents, and thus is less effective; specifically, although those documents (shown in red circle) are negative documents, removing those documents does not help improving the retrieval accuracy for the given query since they are not in the query zone. So, it is very important to ensure the generalized negative language model to be close to **both** *the given query* and *the original negative language model*.

Since the space of all possible negative language models is infinite, we propose two instantiations of our proposed optimization framework to search in a finite space of all feasible language models, which is more tractable.

The first instantiation is based on the KL-divergence distance function where we propose two different methods:

1) Perturbation: to remove the terms (based on the generality measure) in the original negative language model that have less power in pruning non-relevant documents.

2) K-nearest neighborhood (KNN): to search in the neighborhood of the original negative language model to find a more general negative language model.

The second instantiation is based on selecting only powerful terms from the original negative language model where we define an optimization formulation and find an exact solution by solving the optimization formulation directly.

We evaluate the proposed methods by comparing them with the best-performing negative feedback method from the existing work (i.e., MultiNeg) [41] on several representative TREC data sets. Experiment results show that our proposed solutions are more effective in removing non-relevant documents, thus improving the accuracy of search results for difficult queries.

The rest of the paper is organized as follows. We first review related work in section 3. We then describe our optimization framework for generalizing negative language models in section 4. In section 5, we describe the generality notion and instantiations of the proposed optimization framework. In sections 6 and 7, we describe our experimental design and results. Finally, we conclude the paper in section 8.

2. RELATED WORK

The study of difficult queries has attracted much attention recently especially with the launch of ROBUST track in TREC conference which aims at studying the robustness of retrieval models [37, 38]. However, the most effective retrieval models that are developed by ROBUST participants relied on external resources (mostly Web) to perform query expansion which has bypassed the difficulty of the problem in reality. Because there are often no such external resources to exploit and indeed the web resources would not help improve the search accuracy for difficult queries on the Web itself. In this work, we aim at exploiting negative feedback information in the target collection through an optimization framework.

There has been some work on understanding why a query is difficult [4, 6, 15] or identifying difficult queries [38] or on predicating query performance [11, 43] but none of this work has addressed how to improve the search accuracy for difficult queries.

Feedback techniques have proven to be very effective for improving retrieval performance (e.g. [1, 14, 18, 20, 29, 30, 31, 33, 42, 44]). Most feedback methods rely on positive documents, i.e., documents that are judged as relevant to provide useful related terms for query expansion. In contrast, negative (non-relevant) documents have not been found to be very useful. However, there have been some attempts to exploit non-relevant documents; query zone [34] appears to be the only major heuristic proposed to effectively exploit non-relevant information for document routing tasks. It showed that using non-relevant documents that are close to the original query is more effective than using all non-relevant documents in the collection. Also, the work in [27] exploits high-scoring documents outside of top N documents (called pseudo-irrelevant documents) to improve the performance of pseudo-relevance feedback. The work in [40] and later extension to that [41] are the first studies of negative relevance feedback that exploit the top non-relevant documents to improve the ranking of documents. Our work defines an important concept called *generalization of a language model* and we propose an optimization

framework based on this concept to more aggressively (but carefully) prune non-relevant documents, leading to a more effective negative feedback method.

Since we define an optimization framework for negative feedback, our work is also related to previous optimization techniques for pseudo-relevance feedback [7, 8, 12]. Our work is similar to all this work since we also define an optimization framework for term selection. However, it differs in that 1) our optimization framework is defined based on generalizing a negative language model, an important concept that none of the previous work considered; 2) our optimization framework is for negative feedback not for positive feedback, thus helps improving difficult queries.

Our work is also similar to other previous work that consider co-occurrence thesaurus to improve retrieval ranking(e.g., [2, 16, 21, 24, 26, 32, 35]) or hand-crafted thesaurus [22, 39]. Some other work has considered to combine both approaches [5, 23]. Also, the query expansion work in [9] used a term dependency graph in which word co-occurrence was one of the several dependency types. In this paper, we consider word co-occurrence relationship based on Mutual Information [28] (which has shown to be effective for word-to-word translation probabilities in Statistical Translation Language Model [17]) to find the similarity between words in our optimization framework.

3. NEGATIVE FEEDBACK FOR LANGUAGE MODELS

In the following we review basic retrieval model and feedback methods used throughout this paper.

3.1 The KL-Divergence Retrieval Model:

KL-divergence retrieval model [19] is one of the most effective retrieval models in the language modeling framework. This model is a generalization of the query likelihood retrieval model [25] and would score a document D w.r.t query Q based on the negative Kullback-Leibler divergence between the query language model θ_Q and the document language model θ_D:

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

where V is the words in the vocabulary.

Clearly, the two main tasks are to estimate the query language model θ_Q and the document language model θ_D. The document language model θ_D is usually smoothed using Dirichlet prior smoothing which is an effective smoothing method [45].

The query model intuitively captures what the user is interested in, thus would affect retrieval accuracy significantly. The query language model, is often estimated (in case of no feedback) based on $p(w|\theta_Q) = \frac{c(w,Q)}{|Q|}$, where $c(w,Q)$ is the count of word w in query Q and $|Q|$ is the total number of words in the query. Such a model, is not very discriminative because a query is typically extremely short. When there is feedback information, the information would be used to improve our estimate of query language model, θ_Q.

3.2 The Negative Feedback Model

Since a query model described above is usually short, the simple estimation method explained before is not discriminative. Several methods have proposed to improve the estimation of θ_Q by exploiting documents terms, i.e., documents that are used for either relevance or pseudo-relevance feedback [19, 20, 36, 44]. In [44], authors have defined a two-component mixture model (i.e., a fixed background language model, $p(w|\mathcal{C})$, estimated using the whole collection and unknown topic language model to be estimated) and assumed that the feedback documents are generated using such a mixture model.

The basic idea in relevance feedback is to extract useful terms from positive documents and use them to expand the original query. When a query is difficult, it is often impossible to obtain positive (or relevant) documents for feedback. Therefore, the best way would be to exploit the negative documents to perform negative feedback [41]. The idea of negative feedback is to identify distracting non-relevant documents and penalize unseen documents containing such information. More formally:

Given a query Q and a document collection $\mathcal{C}$, a retrieval system returns a ranked list of documents $\mathcal{L}$ where l_i is the $i-$th ranked document in the ranked list $\mathcal{L}$. We assume that the query Q is difficult so that the top n ranked documents (seen so far by the user) are non-relevant. The goal is to study how to use these negative examples, i.e., $\mathcal{N} = \{l_1,\ldots,l_f\}$, to re-rank the next r unseen documents in the original ranked list: $\mathcal{U} = \{l_{f+1},\cdots,l_{f+r}\}$. In our experiments we set $f = 10$ to simulate the first-page result and set $r = 1000$ but any other values could be used without loss of generality.

The two negative feedback methods proposed in [41] are *SingleNeg* and *MultiNeg* methods which we briefly describe below:

SingleNeg: This method adjusts the original relevance score of a document with a single negative model. Let θ_Q and θ_D be estimated query model and document model, respectively. Let θ_N be a negative topic model estimated based on negative feedback documents $\mathcal{N} = \{l_1,\ldots,l_f\}$. The new scoring according to this model is:

$$S(Q,D) = -D(\theta_Q||\theta_D) + \beta.D(\theta_N||\theta_D)$$

In order to estimate θ_N, it is assumed that all non-relevant documents are generated from a mixture model of a unigram language model θ_N and a background language model (generating common words). The log-likelihood of the N sample documents is:

$$L(\mathcal{N}|\theta_N) = \sum_{D\in\mathcal{N}}\sum_{w\in D} c(w,d)log[(1-\lambda)p(w|\theta_N) + \lambda p(w|\mathcal{C})]$$

Where λ (=0.9 in our experiments) is a mixture parameter that controls the weight of the background model, i.e., $p(w|\mathcal{C}) = \frac{c(w,\mathcal{C})}{\sum_w c(w,\mathcal{C})}$. A standard EM algorithm is used to estimate parameters $p(w|\theta_N)$.

MultiNeg: This method adjusts the original relevance scores with multiple negative models. Document D w.r.t query Q is scored as follows:

$$\begin{aligned} S(Q,D) &= S_R(Q,D) - \beta\times S(Q_{neg},D) \\ &= S(Q,D) - \beta\times\max(\bigcup_{i=1}^{f}\{S(Q^i_{neg},D)\}) \\ &= -D(\theta_Q||\theta_D) + \beta\times\max(\bigcup_{i=1}^{f}\{D(\theta_i||\theta_D)\}). \end{aligned}$$

where Q_{neg} is a negative query representation and β is a

parameter that controls the influence of negative feedback. EM algorithm is used to estimate a negative model θ_i for each individual negative document l_i in $\mathcal{N}$. Then we obtain f negative models and combine them with the above formula in our experiments.

Improving negative document language model: A basic component in both SingleNeg and MultiNeg is a negative document language model (i.e., θ_N in SingleNeg and θ_i in MultiNeg), and the accuracy of the estimate of these negative document models may affect the effectiveness of the negative feedback significantly. A main goal of our study is to improve the estimate of a negative document language model through generalizing a basic negative document language model (estimated with an existing approach) with an optimization framework. The improved negative document language models can then be directly plugged into an existing negative feedback method to replace a current negative document language model. In the next section, we present an optimization framework for improving the estimate of negative document language models.

4. AN OPTIMIZATION FRAMEWORK FOR GENERALIZING NEGATIVE LANGUAGE MODELS

4.1 Problem Formulation

Given a query Q and a document collection $\mathcal{C}$, a retrieval system returns a ranked list of documents $\mathcal{L}$ where l_i is the $i-$th ranked document in the ranked list $\mathcal{L}$. We assume that the query Q is difficult so that the top n ranked documents (seen so far by the user) are non-relevant. The goal of our study is to use these negative examples, i.e., $\mathcal{N} = \{l_1, \ldots, l_f\}$ (with language model θ_N) to build a set of more general negative language models, each corresponding to a negative example, so that these general negative language models are better able to describe other unseen negative documents and improve the ranking of documents by pushing down negative documents in the ranked list. More formally, given

$\theta_N = \{\theta_1, \ldots, \theta_f\}$ which is the original negative language model based on documents in $\mathcal{N}$,
where $\theta_1 = \{w_1 : p_1, \ldots, w_m : p_w\}$ (similarly for $\{\theta_2, \ldots, \theta_f\}$), i.e., each language model consists of words along with their probabilities.

Our goal is to estimate $\theta_{G_N} = \{\theta_{G_1}, \ldots, \theta_{Gf}\}$, a set of more general negative language models than θ_N, where each negative language model θ_{G_i} is more general than its corresponding negative language model, θ_i. The improved negative language models θ_{G_N} can then be plugged into a negative feedback method to improve feedback performance.

4.2 Optimization Framework

In order to build a more general negative language model, we propose an *abstract optimization framework* that given θ_N, searches in the space of all language models and finds a set of more general negative language models, i.e., θ_{G_N}. Note that since there are so many general language models, we make it tractable by searching in a *finite space* of all feasible solutions, S.

The objective function is defined as:

$$\theta_{G_N^*} = arg \min_{\theta_{G_N} \in S} (\alpha.\delta(\theta_{G_N}, \theta_N) + (1-\alpha).\delta'(\theta_{G_N}, Q))$$

Subject to:

$$\mathcal{G}(\theta_{G_N^*}) > \mathcal{G}(\theta_N) + \epsilon.$$

This abstract optimization framework defines that we would like to search in the finite space of all language models S, to find a more general negative language model $\theta_{G_N^*}$, that is **1)** close to the original negative language model θ_N (the first term), **2)** and close to query Q (the second term). The closeness to the query ensures the *pruning* power and the closeness to the original negative language model ensures that the feedback model is indeed *accurate*. The generalization constraint is to avoid *over-generalization*. So, we want the general negative language model to deviate by *only* ϵ from its original negative language model (i.e., $\mathcal{G}(\theta_{G_N^*}) > \mathcal{G}(\theta_N) + \epsilon$).

δ and δ' are distance functions. $\mathcal{G}(\theta)$ is a generality measure defined in the next section. α is a tradeoff between closeness to the query and closeness to the original negative language model. ϵ controls the deviation from the original negative language model. These parameters can be optimized based on training data (i.e., cross validation) as done in our experiments.

In order to use the proposed optimization framework for negative feedback, there are three remaining problems to be solved:

1. Definition of the generality measure, i.e., $\mathcal{G}(\theta)$.
2. Definition of the similarity/distance functions δ and δ'.
3. Definition of a search algorithm to efficiently enumerate the most promising candidate language models. Note that the space of potential language models is infinite, but in order to do effective enumeration, our proposed methods search in the finite space (we discuss how to search in the finite space in the next section).

In the next section, we discuss how we solve these problems.

5. INSTANTIATION OF THE OPTIMIZATION FRAMEWORK

5.1 Generalization of Language Models

In this section, we propose to quantify the generality of a language model by introducing a new notion called **Generality**, which is defined as the *expected number* of documents that hit a word. More Formally:

Generality $\mathcal{G}(\theta)$: A language model θ_{G_K} is more general than language model θ_K iff $\mathcal{G}(\theta_{G_K}) > \mathcal{G}(\theta_K)$ where $\mathcal{G}$ is defined as:

$$\mathcal{G}(\theta) = \sum_{w \in \theta} df(w) \times p(w|\theta) \quad (1)$$

Where $df(w)$ is the number of documents containing word w in collection $\mathcal{C}$ (document frequency) and $p(w|\theta)$ is the probability of word w given language model θ. Intuitively, the generality in this formula is captured through both the probability of the word in the language model and the number of documents containing that word in the collection.

Next, we define the distance functions, δ and δ' in our optimization framework, and discuss how to efficiently solve the optimization problem.

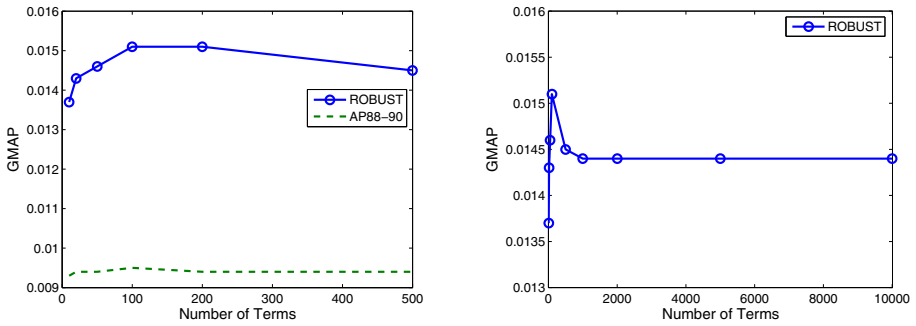

Figure 2: Sensitivity of the number of terms used for negative feedback to GMAP with MultiNeg method.

5.2 Distance Functions δ and δ'

We define two distance functions based on KL-divergence [10] and Term-based-Similarity, respectively. The search strategies vary according to the distance functions.

5.2.1 KL-Divergence

The first instantiation of the abstract optimization framework is to define the distance functions based on KL-divergence. Formally:

$$\delta(\theta_{G_N}, \theta_N) = \frac{1}{2}\left[D_{KL}(\theta_{G_N}||\theta_N) + D_{KL}(\theta_N||\theta_{G_N})\right]$$
$$\delta'(\theta_{G_N}, Q) = D_{KL}(\theta_{G_N}||\theta_Q)$$

where θ_Q is the query language model. Since there might be some terms which are absent in each of those distributions, we use the symmetric version of KL-divergence for the similarity between two distributions θ_N and θ_{G_N}, (i.e. $\delta(\theta_{G_N}, \theta_N)$).

With these instantiations, our objective function is completely defined. So the next challenge is how to efficiently search in the space S to find an optimal solution. Here we propose two strategies:

1. **Perturbation of θ_N**: For each θ_i in the original negative language model, we build a more general negative language model by removing those words w that satisfy $p(w|\theta_i) \times df(w) < \Psi$ and we still ensure that the final negative language model, i.e., θ_{G_i} is still more general than θ_i, i.e., satisfying the generalization constraint and minimizing the objective function. Note that the probabilities are re-normalized to ensure they are comparable.

2. **K-nearest Neighborhood (KNN)**: For this method, we search among the K-neighbors of the negative language model θ_i for those satisfying the generality constraint, and then among the satisfied ones, we select the best one that minimizes our abstract objective function, i.e., it is both close to the original negative language model and query. Then we use the negative language model of that neighbor instead of the original negative language model as a more general negative language model. We denote this as KNN in our experiments.

5.2.2 Term-based Similarity

In this section, we present another instantiation of our abstract optimization framework where we seek an exact solution in a finite space of all language models defined based on term similarity and selection. Specifically, for each top non-relevant document, we get m non-relevant words gained from MultiNeg strategy and feed those terms to the following objective function.

In the abstract optimization framework defined earlier, there are two important components, which we now discuss how to instantiate using term-based similarity. For convenience, in the following we use δ and δ' to denote similarity instead of distance as in the original optimization framework.

1) Closeness to the original negative language model: We instantiate this component as

$$\delta(\theta_{G_N}, \theta_N) = (DF.P)^T \times X$$

where

$DF = [df(w_1), \ldots, df(w_m)]^T$ is a vector of document frequencies (in the collection) for each word w_i (document frequency for word w_i).
$P = [p(w_1|\theta_i), \ldots, p(w_m|\theta_i)]^T$ where i is between zero and 10 for each of the 10 non-relevant documents (e.g., $p(w_1|\theta_i)$ is the probability of word w_1 given the language model θ_i).
$X = [x_1, \ldots, x_m]^T$ is the solution vector, which tells which of the words among m words should be included as final words according to the objective function.

2) Closeness to query:

$$\delta'(\theta_{G_N}, Q) = Sim_Q^T \times X$$

where $Sim_Q = [Sim(w_1, q), \ldots, Sim(w_m, q)]^T$ which is the similarity between each word and query word. In case a query consists of multiple words, we get their average similarity. We define the co-occurrence between two words as their similarity. The co-occurrence similarity is based on Mutual Information [28]. Mutual Information is a good measure to assess how two words are related. We compute the mutual information scores for each pair of two words w and u in the collection. Informally, mutual information compares the probability of observing w and u together (the joint probability) with the probabilities of observing w and u independently. The mutual information between words w and u is calculated as follows:

$$I(w;u) = \sum_{X_w=0,1} \sum_{X_u=0,1} p(X_w, X_u) \log \frac{p(X_w, X_u)}{p(X_w)p(X_u)} \quad (2)$$

where X_u and X_w are binary variables indicating whether u or w is present or absent.

And,

$$\begin{aligned}
p(X_u = 1) &= \frac{c(X_u = 1)}{N} \\
p(X_u = 0) &= 1 - p(X_u = 1) \\
p(X_w = 0, X_u = 0) &= 1 - p(X_w = 0, X_u = 1) \\
p(X_w = 1, X_u = 1) &= \frac{c(X_w = 1, X_u = 1)}{N} \\
p(X_w = 1, X_u = 0) &= \frac{(c(X_w = 1) - c(X_w = 1, X_u = 1))}{N} \\
p(X_w = 0, X_u = 1) &= \frac{(c(X_u = 1) - c(X_w = 1, X_u = 1))}{N} \\
p(X_w = 0, X_u = 0) &= 1 - p(X_w = 0, X_u = 1) \\
&\quad -p(X_w = 1, X_u = 0) - p(X_w = 1, X_u = 1)
\end{aligned}$$

The objective function is then defined as follows:

Maximize

$$((DF.P)^T \times X + \gamma . Sim_Q^T \times X)$$

Subject to:

$$\mathbf{C1}: 0 \leq x_i \leq 1$$
$$\mathbf{C2}: \textstyle\sum_i x_i = 1$$

Figure 3: Optimization Formulation

Since the two components defined in this optimization framework (i.e., closeness to query and closeness to original negative language model) are not really comparable, we re-parameterize it as above, i.e., α in the abstract optimization framework is replaced with γ parameter when we instantiate it as shown in the figure.

Our solution, i.e., X would tell us which terms should be added to the final negative language model. $x_i = 1$, if the word should strongly be selected and zero otherwise. In all other cases, it is between 0 and 1. So, it can serve as our confidence in selecting the terms. We can then recover the θ_{G_N} based on X as follows (re-normalization is done):

$$p(w_i|\theta_{G_N}) = x_i . p(w_i|\theta_N)$$

We denote this method as OptMultiNeg in our experiment results.

6. EXPERIMENT DESIGN

6.1 Data Sets

We experiment with two data sets that are representative of heterogeneous and homogeneous data sets, respectively. Our first data set is ROBUST track of TREC 2004 which has 528,155 news articles [38]. On average, each document has 521.89 words and there are 249 queries[1] in this set. The robust track is a standard ad hoc retrieval with an emphasis on the overall reliability of IR systems which contains difficult queries and is a **heterogeneous** data set. The data set is called "ROBUST".

The second data set is the AP88-90 in ad hoc retrieval which is a **homogeneous** data set. It contains 242,918 documents. On average, each document has about 464.226 terms. We used queries 51−200 for our experiments. The data set is called "AP88-90".

[1]One query was dropped because the evaluators did not provide any relevant documents for it.

For both data sets, preprocessing of documents and queries is minimum and involves only stemming with Porter stemmer but without removing any stopwords. As in some previous work (e.g., [45]), we did not remove stop words for two reasons: (1) A robust model should be able to discount the stop words. (2) Removing stop words would introduce one extra parameter (e.g. the number of stop words) into our experiments.

Since our goal is to study difficult queries, we consider naturally difficult queries from our data sets as follows:

We follow the definition of naturally difficult queries as in [41]. A query is naturally difficult when its P@10=0 given a retrieval model. For language model (LM), we use the standard ranking function (i.e., KL-divergence retrieval model with Dirichlet Prior Smoothing [19]) to select their naturally difficult queries. We first optimize the parameter μ (Dirichlet Prior) for LM on all data sets. The optimal is gained when $\mu = 2000$ for ROBUST data set and $\mu = 3000$ for AP88-90 data set using Lemur toolkit[2]. We then fix these parameters in all the following experiments. Using the optimal parameter setting, we select those queries whose P@10=0 as our naturally difficult queries. 26 queries in ROBUST and 38 queries from AP88-90 are selected as naturally difficult queries and we experiment with these query sets in the rest of the paper.

6.2 Baselines and Experiment Procedure

Since previous work has shown MultiNeg is the most effective negative feedback method [41], we focus on applying the proposed language model generalization method to MultiNeg, and compare our proposed methods (i.e., OptMultiNeg, KNN and Perturbation) with two state of the art methods proposed in [41], i.e., SingleNeg and MultiNeg. All our proposed solutions follow the same scoring as in [41], i.e., each negative document is used to penalize unseen documents, however the language models in our methods are more general. All the experiments are done using Lemur toolkit. The exact solution for OptMultiNeg method is solved using MATLAB.

Our experiment setup is the same as the one adopted in [41]. The goal is to simulate a scenario when a user has found the top-K ranked documents are non-relevant (i.e., these document were skipped by a user without being viewed) and is about to view the rest of the search results. At this point, we can naturally apply negative feedback to re-rank all the *unseen* documents. As in [41], we set $K = 10$, which simulates the scenario of applying negative feedback when a user has not found any relevant document on the first page of search results and is about to view the next page of results (a realistic assumption in the case of difficult topics).

With this setup, the top-ranked 1000 **unseen** documents for all runs were compared in terms of two sets of performance measures: (1) Mean Average Precision (MAP) and Geometric mean Average Precision (GMAP), which serve as good measures for the overall ranking accuracy. (2) Mean Reciprocal Rank (MRR) and Precision at 10 (P@10), which reflect the utility from users perspective who only read the top-ranked documents. Please note that since we are working with difficult queries, GMAP is considered as our **main measure**, however, we show our experimental results based on all measures for the sake of completeness.

In order to set two baseline parameters (i.e., β and ρ), we do cross validation as follows: We fix the number of feedback terms to 100 and learn two baseline parameters, i.e., β (described in section 3, i.e., a parameter to control the in-

[2]http://www.lemurproject.org/

Table 1: Performance of the Optimization framework on ROBUST data set based on cross validation (left) and Upper bound (right), * and + means improvements over MultiNeg and SingleNeg are statistically significant with Wilcoxon signed-rank test, respectively. We only show the significance tests for GMAP measure since this is our main measure given that we are improving the performance of difficult queries. These results are only based on 100 words extracted from each top non-relevant document.

Methods	MAP	**GMAP**	MRR	P@10
SingleNeg (baseline)	0.0321	0.0134	0.1775	0.088
MultiNeg (baseline)	0.0318	0.0132	0.2124	0.08
KNN	0.0322	0.0138^{*+}	0.2137	0.084
Perturbation	0.0327	0.0136^{*}	0.2597	0.088
OptMultiNeg	0.0365	$\mathbf{0.0144}^{*+}$	0.2804	0.084
OptMultiNeg/MultiNeg	14.7%	9%	32%	5%
OptMultiNeg/SingleNeg	13.7%	7%	57%	-4.5%
Perturbation/MultiNeg	2.8%	3%	22%	10%
Perturbation/SingleNeg	1.8%	1.5%	46%	0%
KNN/MultiNeg	1.2%	4.5%	0.6%	5%
KNN/SingleNeg	0.31%	3%	20%	-4.5%

Methods	MAP	**GMAP**	MRR	P@10
SingleNeg (baseline)	0.0351	0.0145	0.2195	0.092
MultiNeg (baseline)	0.0361	0.0151	0.2388	0.088
KNN	0.034	0.0154^{*+}	0.2445	0.088
Perturbation	0.0393	$\mathbf{0.0159}^{*+}$	0.3407	0.1
OptMultiNeg	0.0378	$\mathbf{0.0156}^{*+}$	0.3223	0.088

Table 2: Performance of the Optimization framework on AP88-90 data set based on cross validation (left) and Upper bound (right), * and + means improvements over MultiNeg and SingleNeg are statistically significant with Wilcoxon signed-rank test, respectively. We only show the significance tests for GMAP measure since this is our main measure given that we are improving the performance of difficult queries. These results are only based on 100 words extracted from each top non-relevant document.

Methods	MAP	**GMAP**	MRR	P@10
SingleNeg (baseline)	0.0377	0.0091	0.1500	0.0631
MultiNeg (baseline)	0.0389	0.0093	0.1521	0.0651
KNN	0.0388	0.0094^{+}	0.1554	0.0657
Perturbation	0.0386	0.0093^{+}	0.1523	0.0631
OptMultiNeg	0.0391	$\mathbf{0.0096}^{*+}$	0.1937	0.0658
OptMultiNeg/MultiNeg	0.5%	3%	27%	1.1%
OptMultiNeg/SingleNeg	3.7%	5.5%	29%	4%
Perturbation/MultiNeg	-0.7%	0%	0.13%	-3%
Perturbation/SingleNeg	2.4%	2%	1.5%	0%
KNN/MultiNeg	-0.25%	1%	2.2%	0.9%
KNN/SingleNeg	2.9%	3%	3.6%	4.1%

Methods	MAP	**GMAP**	MRR	P@10
SingleNeg (baseline)	0.0381	0.0092	0.1517	0.0684
MultiNeg (baseline)	0.0396	0.0094	0.1914	0.0684
KNN	0.0393	0.0095^{+}	0.1755	0.0684
Perturbation	0.0403	0.0096^{*+}	0.197	0.071
OptMultiNeg	0.0392	$\mathbf{0.0097}^{*+}$	0.1950	0.0658

fluence of the negative feedback) and "number of documents to penalize" (ρ in [41]) based on the training data. Since there are not so many naturally difficult queries in TREC data sets, we do leave-one-out cross validation to learn the parameters for the baselines. The parameters of our proposed methods, i.e., γ, α, Ψ and ϵ are also leaned through leave-one-out cross validation. Thus, the parameters of all the methods (i.e., our proposed methods and baseline methods) are optimized in the same way (i.e., leave-one-out cross validation) to have a fair comparison. Please note that the number of feedback terms is chosen to be 100 (for each document) without loss of generality. As shown in Figure 2, when the number of feedback terms are small (smaller than 100), the performance is not good (with MultiNeg baseline method) and when the number of feedback terms are very large (larger than 200), the performance drops. To aid exposition, we increase the number of feedback terms to even 10000 in Figure 2 (right), and as shown, the performance drops when the number of feedback terms increases.

We also experiment to get the optimal performance on test queries. For that, we vary β from 0.1 to 0.9 and ρ from 50 to 1000 on test queries and select the best-performing set of parameters (i.e., one β and one ρ for all test queries) according to GMAP measure as done in [41][3] but our main focus would be on the results gained from cross validation.

[3]The authors only reported the upper bound results (or the optimal results) without reporting the results based on cross validation. However, one should note that parameters should be learned based on training data sets.

We also vary γ, α, Ψ and ϵ in our proposed solutions as shown in Figures 4 and 5 to get the optimal parameters.

7. EXPERIMENT RESULTS

7.1 Effectiveness of our Proposed Solutions

In order to see the effectiveness of our proposed solutions, we compare them with the baselines methods.

The results are shown in Tables 1 and 2 based on both collections ROBUST and AP88-90, respectively. Table 1 (left) shows the cross validation results on ROBUST data set and Table 1 (right) shows the upper bound results. The results in Table 2 (left) and (right) also show cross validation and upper bound results based on AP88-90 data set, respectively. The upper bound baseline results are comparable to their corresponding results reported previously [41] (the authors of [41] did not report the cross validation results). From these two tables, we have the following observations:

(1) The results both based on upper bound and cross validation show that our proposed methods outperform the baselines in terms of GMAP (since this serves as our main measure and we maximize based on this measure when learning the parameters). For example, on ROBUST data set, the OptMultiNeg method can improve GMAP from 0.0132 (in MultiNeg method) to 0.0144, about 9% relative improvement which is a significant improvement given that the difficult queries are harder to be improved. On AP88−90, our proposed methods can also significanlty improve over baseline methods. The results are also statistically significant

Table 3: Performance of the Optimization framework on ROBUST data set based on cross validation (left) and Upper bound (right), * and + means improvements over MultiNeg and SingleNeg are statistically significant with Wilcoxon signed-rank test, respectively. We only show the significance tests for GMAP measure. These results are based on all words extracted from each top non-relevant document.

Methods	MAP	GMAP	MRR	P@10
SingleNeg (baseline)	0.0336	0.0137	0.1861	0.088
MultiNeg (baseline)	0.0287	0.0127	0.1502	0.08
OptMultiNeg	0.0316	**0.0141**$^{*+}$	0.2082	0.08

Methods	MAP	GMAP	MRR	P@10
SingleNeg (baseline)	0.0350	0.0147	0.2119	0.088
MultiNeg (baseline)	0.0318	0.0144	0.2147	0.088
OptMultiNeg	0.0326	**0.0151**$^{*+}$	0.2463	0.088

Table 4: Performance of the Optimization framework on AP88-90 data set based on cross validation (left) and Upper bound (right), * and + means improvements over MultiNeg and SingleNeg are statistically significant with Wilcoxon signed-rank test, respectively. We only show the significance tests for GMAP measure. These results are based on all words extracted from each top non-relevant document.

Methods	MAP	GMAP	MRR	P@10
SingleNeg (baseline)	0.0395	0.0094	0.1702	0.0710
MultiNeg (baseline)	0.0386	0.0091	0.1723	0.0684
OptMultiNeg	0.0389	**0.0096**$^{*+}$	0.1869	0.0710

Methods	MAP	GMAP	MRR	P@10
SingleNeg (baseline)	0.0396	0.0095	0.1752	0.071
MultiNeg (baseline)	0.0401	0.0096	0.1981	0.071
OptMultiNeg	0.0396	**0.0097**$^{*+}$	0.1917	0.0763

based on Wilcoxon signed-rank tests (for GMAP measure) for those cases marked in the tables. We also show the percentage improvement for our methods over the baselines (based on cross validation only) in Tables 1 and 2 (left). For example, OptMultiNeg/MultiNeg means the improvement of OptMultiNeg over MultiNeg baseline method. Please note that we only show the percentage improvement for cross validation results *only* since these are our main results.

(2) Another interesting observation is that although we only maximize based on GMAP, the results of our proposed methods also outperform MAP and MRR measure in most cases (cross validation results).

(3) Comparing our proposed methods shows that the *OptMultiNeg* method outperforms (according to GMAP measure) all the other proposed methods since it finds an exact optimal solution.

Given that these are all very difficult queries where the state of the art retrieval models worked poorly and negative feedback can be done automatically based on implicit feedback information without requiring any additional user effort, these results are quite encouraging.

Table 5 shows some sample words along with their probabilities selected both based on MultiNeg and OptMultiNeg for TREC query 690 (This is an example where our OptMultiNeg outperformed the baselines in terms of GMAP measure a lot). We only show top 10 selected terms and also calculate their generality measures for these two methods. It is shown that our OptMultiNeg method is more general (i.e., $\mathcal{G}_{10}(\theta') > \mathcal{G}_{10}(\theta)$). Please note that the generality values calculated here is based on only 10 shown words; we should also mention that $\mathcal{G}_{100}(\theta') > \mathcal{G}_{100}(\theta)$, i.e., our OptimizeMultiNeg is more general than MultiNeg when top 100 words are selected (this is the number of feedback terms that we experimented with). In addition, we show another example (TREC query 343) where the performance was hurt a lot. The sample words selected are shown in Table 6 which shows the over-generalization both in terms of the words themselves (e.g., words are not specific compared to the original words) and in terms of the generalization value.

Another experiment setup is to give all the words extracted from the original non-relevant documents to the OptMultiNeg method and let the method choose among them (instead of choosing among top 100 words). Our hypothesis is that the results of our OptMultiNeg from this experiment should also be better than that of baselines for the case when we have only 100 terms. The results of such experiments are shown in Tables 3 and 4 for both data sets. As shown in the tables, the results are worse than their corresponding results in Tables 1 and 2 (as we expected from the analyses shown in Figure 2). However, the OptMultiNeg still outperforms the baselines which confirms our hypothesis.

Efficiency of our Proposed Methods: Since we only have 10 negative examples (first-page result simulation), building a language model for each of them can be done efficiently online. Also, since we search in the finite space of all the potential language models, the search is fast enough to be done online, i.e., even for the OptMultiNeg method, search does not take a lot of time to select words since the space of the selection is also finite.

Table 5: 10 Sample word selected from MultiNeg model (left) and OptMultiNeg (right) for Query TREC 690= "colleg educ advantag". Note that words are stemmed.

q	$p(q\|\theta)$
shanghai	0.0512
school	0.0354
we	0.0336
reform	0.0171
teacher	0.0155
system	0.0141
higher	0.0138
establish	0.0125
learn	0.0124
graduat	0.0117
$\mathcal{G}_{10}(\theta) = 32,899$	

q	$p(q\|\theta')$
shanghai	0.275
we	0.1804
reform	0.0921
system	0.0760
establish	0.0672
percent	0.0542
cooper	0.0450
must	0.0444
construct	0.0441
develop	0.0294
$\mathcal{G}_{10}(\theta') = 33,461$	

7.2 Parameter Sensitivity Study

Since there are parameters associated with our proposed methods, in this section, we analyze the sensitivity of the parameters to GMAP measure.

Figure 4 (left) shows sensitivity of γ to GMAP in OptMultiNeg method for both ROBUST and AP88-90 data sets. We first fix β, ρ to their optimal values and vary γ to see how it changes to the GMAP value. As shown in the figure, the two-end points show poor performance of the OptMultiNeg method which indicates that over-generalization hurts the performance.

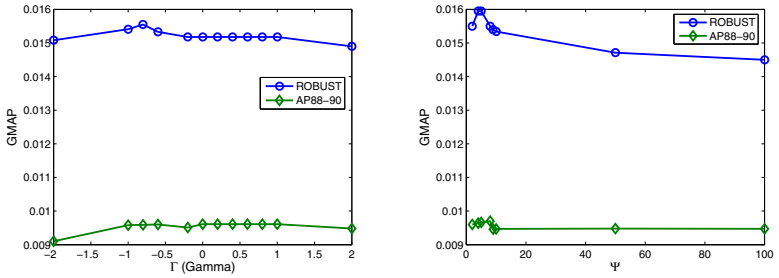

Figure 4: Sensitivity of different parameters to GMAP measure: γ in OptMultiNeg method to GMAP (left), Ψ in Perturbation method to GMAP (right).

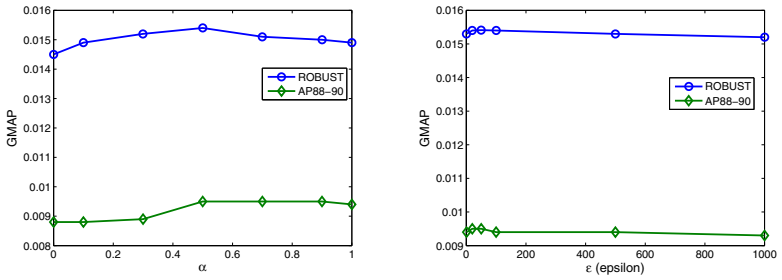

Figure 5: Sensitivity of different parameters to GMAP measure: α in KNN method to GMAP (left), ϵ in KNN method to GMAP (right).

Table 6: 10 Sample word selected from MultiNeg model (left) and OptMultiNeg (right) for Query TREC 343= "polic death". It is an example that over-generalization hurts the performance. Note that words are stemmed.

q	$p(q\|\theta)$
injuri	0.0336332
anc	0.0318221
attack	0.0264466
support	0.0204667
hostel	0.0200951
incid	0.0193305
ifp	0.0180334
allegedli	0.0158444
secur	0.0141943
member	0.0140543
$\mathcal{G}_{10}(\theta) = 10,990$	

q	$p(q\|\theta')$
anc	0.1325
support	0.0852
secur	0.0591
member	0.0585
forc	0.0497
were	0.0479
week	0.0402
sub	0.04023
11	0.0323
region	0.0317
$\mathcal{G}_{10}(\theta') = 20,850$	

Figure 4 (right) shows the sensitivity of Ψ in Perturbation method. As shown in the figure, when we increase the amount of Ψ, it hurts the performance, so it indicates that over-generalization hurts the performance and Ψ should be set to: $1 < \Psi < 5$ to ensure improvement in the performance.

Figure 5 (left) also shows the sensitivity of parameter α in KNN method[4] which is a tradeoff between closeness to query and closeness to negative language model. The best performance is gained when $0.4 < \alpha < 0.6$ which confirms our hypothesis that the general negative language model should be close to both query and original negative language model.

Figure 5 (right) also shows the sensitivity of parameter ϵ in KNN method. As we increase ϵ, the performance hurts indicating that over-generalization hurts the performance and ϵ should be set to: $1 < \epsilon < 50$.

[4]The same pattern can also be seen with Perturbation method.

8. CONCLUSIONS AND FUTURE WORK

How to help users when their queries do not work well with state of the art retrieval methods is a very important, yet difficult challenge. Negative feedback is an important and useful technique for tackling this challenge, and can be done automatically without requiring extra user effort based on implicit feedback information (such as skipping all the results on the first page and attempting to view additional results on the next page). In this paper, we addressed the problem of data sparseness in negative feedback in the language modeling framework by proposing an abstract optimization framework, in which we learn from a few top-ranked non-relevant examples, and search in the space of all candidate language models to build a more general negative language model. This general negative language model has been shown to have more power in pruning the non-relevant documents on two representative TREC data sets, outperforming state of the art negative feedback methods significantly for difficult queries. The proposed method is general and can be potentially implemented in any search engine applications to improve a user's experience in the case of difficult topics.

Our work is only a first step in the exploration of the general idea of generalization of language models. There are many interesting directions to further explore. First, we can relax the feedback condition to include a small number of positive feedback examples, which would lead to the interesting problem of how to generalize both negative and positive language models simultaneously. Second, it would be interesting to apply Explanation-based Learning (EBL) [13] to feedback and construct a generalized language model based on a formal explanation of why a document has been judged as non-relevant (or relevant). Finally, we believe that the idea of generalizing language models can also be potentially

applied to many other tasks in interactive retrieval where we need to infer a user's intent based on the user's behavior.

9. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-0713581, CNS-0834709, and CNS 1028381, by NIH/NLM grant 1 R01 LM009153-01, and by a Sloan Research Fellowship. Maryam Karimzadehgan was supported by the Google PhD fellowship. Any opinions, findings, conclusions, or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

10. REFERENCES

[1] R. Attar and A. S. Fraenkel. Local feedback in full-text retrieval systems. *J. ACM*, pages 24(3):397–417, 1977.

[2] J. Bai, D. Song, P. Bruza, J. Y. Nie, and G. Cao. Query expansion using term relationships in language models for information retrieval. *ACM CIKM*, pages 688–695, 2005.

[3] C. Buckeley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec3. *TREC94*, pages 69–80, 1994.

[4] C. Buckley. Why current ir engines fail. *ACM SIGIR*, pages 584–585, 2004.

[5] G. Cao, J. Y. Nie, and J. Bai. Integrating word relationships into language models. *ACM SIGIR*, pages 298–305, 2005.

[6] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? *ACM SIGIR*, pages 390–397, 2006.

[7] K. Collins-Thompson. Estimating robust query models using convex optimization. *NIPS*, 2008.

[8] K. Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. *ACM CIKM*, pages 837–846, 2009.

[9] K. Collins-Thompson and J. Callan. Query expansion using random walk models. *ACM CIKM*, pages 704–711, 2005.

[10] T. Cover and J. Thomas. Elements of information theory. *John Wiley and Sons, New York, USA*, 1991.

[11] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. *ACM SIGIR*, pages 299–306, 2002.

[12] J. V. Dillon and K. Collins-Thompson. A unified optimization framework for robust pseudo-relevance feedback algorithms. *ACM CIKM*, pages 1069–1078, 2010.

[13] T. Ellman. Explanation-based learning: a survey of programs and perspectives. *ACM Comput. Surv.*, 21:163–221, June 1989.

[14] D. Harman. Relevance feedback revisited. In *Proceedings of ACM SIGIR 1992*, pages 1–10, 1992.

[15] D. Harman and C. Buckley. Sigir 2004 workshop: Ria and where can ir go from here. *SIGIR Forum*, pages 38(2):45–49, 2004.

[16] Y. Jing and B. Croft. An association thesaurus for information retrieval. *RIAO*, pages 141–160, 1994.

[17] M. Karimzadehgan and C. Zhai. Estimation of statistical translation models based on mutual information for ad hoc information retrieval. *ACM SIGIR*, pages 323–330, 2010.

[18] M. Karimzadehgan and C. Zhai. Exploration-exploitation tradeoff in interactive relevance feedback. *ACM CIKM*, pages 1397–1400, 2010.

[19] J. Laffetry and C. Zhai. Document language models, query models, and risk minimization for information retrieval. *ACM SIGIR*, pages 111–119, 2001.

[20] V. Lavrenko and W. B. Croft. Relevance-based language models. *SIGIR*, pages 120–127, 2001.

[21] M. Lesk and B. Croft. Word-word associations in document retrieval systems. *American Documentation*, 20:20–27, 1969.

[22] S. Liu, F. Lin, C. Yu, and W. Meng. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. *ACM SIGIR*, pages 266–272, 2004.

[23] R. Mandala, T. tokunaga, H. Tanaka, and K. Satoh. Ad hoc retrieval experiments using wordnet and automatically constructed thesauri. *TREC-7*, pages 475–481, 1998.

[24] H. J. Peat and P. Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. *J. of Information science*, 42(5):378–383, 1991.

[25] J. Ponte and W. Croft. A language modeling approach to information retrieval. *ACM SIGIR*, 1998.

[26] Y. Qiu and H. Frei. Concept based query expansion. *ACM SIGIR*, pages 160–169, 1993.

[27] K. Raman, R. Udupa, P. Bhattacharya, and A. Bhole. On improving pseudo-relevance feedback using pseudo-irrelevant documents. *LNCS*, pages 573–576, 2010.

[28] C. J. V. Rijsbergen. Information retrieval. *Butterworths*, 1979.

[29] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society of Information Science*, pages 27(3):129–146, 1976.

[30] J. Rocchio. Relevance feedback in information retrieval. *In the SMART Retrieval System*, pages 313–323, 1971.

[31] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of Information Science*, pages 41(4):288–297, 1990.

[32] H. Schutze and J. O. Pedersen. A co-occurrence based thesaurus and two applications to information retrieval. *Information and processing management*, 33(3):307–318, 1997.

[33] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. *ACM SIGIR*, pages 43–50, 2005.

[34] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. *ACM SIGIR*, pages 25–32, 1997.

[35] A. F. Smeaton and C. J. V. Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):239–246, 1983.

[36] T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. *ACM SIGIR*, pages 162–169, 2006.

[37] E. M. Voorhees. Overview of the trec 2004 robust retrieval track. *TREC*, 2004.

[38] E. M. Voorhees. Draft: Overview of the trec 2005 robust retrieval track. *Notebook of TREC2005*, 2005.

[39] E. M. Voorhess. Query expansion using lexical-semantic relations. *ACM SIGIR*, pages 61–69, 1994.

[40] X. Wang, H. Fang, and C. Zhai. Improve retrieval accuracy for difficult queries using negative feedback. *ACM CIKM*, pages 991–994, 2007.

[41] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. *ACM SIGIR*, pages 219–226, 2008.

[42] J. Xu and W. B. Croft. Query expansion using local and global document analysis. *ACM SIGIR*, pages 4–11, 1996.

[43] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. *ACM SIGIR*, pages 512–519, 2005.

[44] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. *ACM CIKM*, pages 403–410, 2001.

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