A Study of Methods for Negative Relevance Feedback

Xuanhui Wang
University of Illinois at
Urbana-Champaign
Urbana, IL 61801
xwang20@cs.uiuc.edu

Hui Fang
The Ohio State University
Columbus, OH 43210
hfang@cse.ohio-state.edu

ChengXiang Zhai
University of Illinois at
Urbana-Champaign
Urbana, IL 61801
czhai@cs.uiuc.edu

ABSTRACT

Negative relevance feedback is a special case of relevance feedback where we do not have any positive example; this often happens when the topic is difficult and the search results are poor. Although in principle any standard relevance feedback technique can be applied to negative relevance feedback, it may not perform well due to the lack of positive examples. In this paper, we conduct a systematic study of methods for negative relevance feedback. We compare a set of representative negative feedback methods, covering vector-space models and language models, as well as several special heuristics for negative feedback. Evaluating negative feedback methods requires a test set with sufficient difficult topics, but there are not many naturally difficult topics in the existing test collections. We use two sampling strategies to adapt a test collection with easy topics to evaluate negative feedback. Experiment results on several TREC collections show that language model based negative feedback methods are generally more effective than those based on vector-space models, and using multiple negative models is an effective heuristic for negative feedback. Our results also show that it is feasible to adapt test collections with easy topics for evaluating negative feedback methods through sampling.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Retrieval models

General Terms: Algorithms

Keywords: Negative feedback, difficult topics, language models, vector space models

1. INTRODUCTION

No retrieval model is able to return satisfactory results for every query. Indeed, a query might be so difficult that a large number of top-ranked documents are non-relevant. In such a case, a user would have to either reformulate the query or go far down on the ranked list to examine more documents. Thus studying how to improve search results for such difficult topics is both theoretically interesting and practically important.

A commonly used strategy to improve search accuracy is through feedback techniques, such as relevance feedback [12, 8], pseudo-relevance feedback [1, 20], and implicit feedback [14]. In the case of a difficult topic, we likely will have only negative (i.e., non-relevant) examples, raising the important question of how to perform relevance feedback with only negative examples. We refer to this problem as *negative feedback*. Ideally, if we can perform effective negative feedback, when the user could not find any relevant document on the first page of search results, we would be able to improve the ranking of unseen results in the next a few pages.

However, whether such negative feedback can indeed improve retrieval accuracy is still largely an open question. Indeed, the effectiveness of current feedback methods often rely on relevant documents; negative information, such as non-relevant documents, is mostly ignored in past work [4, 8].

On the surface, any standard relevance feedback technique can be applied to negative relevance feedback. However, our recent work [19] has shown that special care and special heuristics are needed to achieve effective negative feedback. Specifically, in this work, we have shown that some language model-based feedback methods, although quite effective for exploiting positive feedback information, cannot naturally handle negative feedback, thus several methods were proposed to perform negative feedback in language modeling framework. However, this study is neither comprehensive nor conclusive for several reasons: (1) It is only limited to language models; vector space models have not been evaluated. (2) The results are evaluated over only one collection. (3) The lack of systematic experiment design and result analysis makes it hard to know the advantages or disadvantages of different methods.

In this paper, we conduct a more systematic study of different methods for negative relevance feedback. Our study is on two representative retrieval models: vector space models and language models. We first categorize negative feedback techniques into several general strategies: single query model, single positive model with single negative query model, and single positive model with multiple negative query models. Following these strategies, we then develop a set of representative retrieval methods for both retrieval models. Systematic comparison and analysis are conducted on two large representative TREC data sets. Ideally, test sets with sufficient naturally difficult topics are required to evaluate these negative feedback methods, but there are not many naturally difficult topics in the existing TREC data collections. To overcome this difficulty, we use two sampling strategies to adapt a test collection with easy topics to evaluate negative feedback. The basic idea of our sampling methods is to simulate difficult queries from easy ones through deleting a set of relevant documents so that the results become poor. The effectiveness of these sampling methods is also verified on the TREC data sets.

Our systematic study leads to several interesting conclusions. We find that language model-based negative feedback methods are

SIGIR'08, July 20–24, 2008, Singapore.
Copyright 2008 ACM 978-1-60558-164-4/08/07 ...$5.00.

generally more effective and robust than those based on vector space models possibly due to more accurate learning of negative models. While cluster hypothesis [7] generally holds for relevant documents, our results show that negative documents do not cluster together. Thus adapting standard relevance feedback to learn a single query model is not optimal for negative feedback, and using multiple negative models is more effective than a single negative model since negative documents may distract in different ways. Our results also show that it is feasible to adapt test collections with easy topics (through sampling) to evaluate negative feedback methods.

The rest of the paper is organized as follows. In Section 2, we review the related work. In Section 3, we describe our problem setup and different techniques for negative feedback. We describe our sampling methods to simulate difficult topics by adapting easy ones in Section 4. Experiments are analyzed and discussed in Section 5. We conclude this paper and discuss our future work in Section 6.

2. RELATED WORK

The study of difficult queries has attracted much attention recently, partly due to the launching of the ROBUST track in the TREC conference, which aims at studying the robustness of a retrieval model and developing effective methods for difficult queries [18, 17]. However, the most effective methods developed by the participants of the ROBUST track tend to rely on external resources (notably the Web) to perform query expansion, which has in some sense bypassed the difficulty of the problem as in reality, there is often no such external resource to exploit, or otherwise, the user would have directly gone to the external resource to find information. Indeed, the Web resource would not help improve search accuracy for difficult topics on the Web itself. In our work, we aim at exploiting negative feedback information in the target collection from which we want to retrieve information.

There has been some work on understanding why a query is difficult [6, 3, 2], on identifying difficult queries [17], and on predicting query performance [21]. But none of this work has addressed the important question of how to improve search accuracy for difficult queries.

Feedback techniques have been extensively studied and mostly shown to be effective to improve retrieval accuracy [12, 10, 1, 13, 5, 20, 22, 14]. In general, most feedback techniques rely on positive documents – documents that are explicitly judged as relevant or implicitly assumed to be relevant – to provide useful related terms for query expansion. In contrast, negative (i.e., non-relevant) documents have not been found to be very useful. In general, exploiting non-relevant information is largely unexplored; query zone [16] appears to be the only major heuristic proposed to effectively exploit non-relevant information in document routing tasks. It showed that using non-relevant documents which are close to the original queries is more effective than using all non-relevant documents in the whole collection. However, this problem was studied for document routing tasks and a lot of relevant documents are used. Our problem setting is quite different in that we only have non-relevant documents for feedback, and we start with non-relevant documents close to a query to study how to use this negative information optimally in ad hoc retrieval.

Our recent work [19] is the first study on the problem of negative feedback in language models. It shows that special techniques are needed to handle negative feedback. Our current work can be regarded as an extension of this previous work to include additional retrieval models, additional heuristics, and making more conclusive findings. The main differences between our current work and the previous work [19] include: (1) We extend previous study and propose several *general* negative feedback strategies that can be applied to both vector space and language models. (2) We study two sampling methods to construct larger collections to evaluate negative feedback methods. (3) Our experiments are more systematic and conducted over more collections.

3. NEGATIVE FEEDBACK TECHNIQUES

3.1 Problem Formulation

We formulate the problem of negative feedback in a similar way as presented in [19]. Given a query Q and a document collection $\mathcal{C}$, a retrieval system returns a ranked list of documents $\mathcal{L}$. L_i denotes the i-th ranked document in the ranked list. We assume that Q is so difficult that all the top f ranked documents (seen so far by a user) are non-relevant. The goal is to study how to use these negative examples, i.e., $\mathcal{N} = \{L_1, ..., L_f\}$, to rerank the next r unseen documents in the original ranked list: $\mathcal{U} = \{L_{f+1}, ..., L_{f+r}\}$. We set $f = 10$ to simulate that the first page of search results are irrelevant, and set $r = 1000$. We use the following notations in the rest of the paper:

$S(Q, D)$ is the relevance score of document D for query Q.
$c(w, D)$ is the count of word w in document D.
$c(w, Q)$ is the count of word w in query Q.
$|\mathcal{C}|$ is the total number of documents in the collection $\mathcal{C}$.
$df(w)$ is the document frequency of word w.
$|D|$ is the length of document D.
$avdl$ is the average document length.
$\mathcal{N}$ is the set of negative feedback documents.
$\mathcal{U}$ is the set of unseen documents to be reranked.

3.2 General Strategies for Negative Feedback

3.2.1 Query Modification

Since negative feedback can be regarded as a special case of relevance feedback where no positive example is available, our first general strategy is simply to apply any existing feedback methods (e.g., Rocchio [12]) to use only non-relevant examples. We call this strategy *query modification* because most existing feedback methods would achieve feedback through modifying the representation of a query based on relevant and non-relevant feedback documents. In effect, they often introduce additional terms to expand a query and assign more weight to a term with more occurrences in relevant documents and less weight or negative weight to a term with more occurrences in non-relevant documents.

Some existing feedback methods, such as Rocchio method [12], already have a component for using negative information, so they can be directly applied to negative feedback. However, other methods, such as model-based feedback methods in language modeling approaches [22], can not naturally support negative feedback, thus extension has to be made to make them work for negative feedback [19]. Later we will further discuss this.

Note that with this strategy, we generally end up with one *single* query model/representation which combines both positive information from the original query and negative information from the feedback documents.

3.2.2 Score Combination

The query modification strategy mixes both positive and negative information together in a single query model. Sometimes it is not natural to mix these two kinds of information as in the case of using generative models for feedback [22]. A more flexible alternative strategy is to maintain a positive query representation and a negative query representation *separately*, and combine the scores of a document w.r.t. both representations. We call this stategy *score combination*.

With this strategy, negative examples can be used to learn a negative query representation which can then be used to score a document based on the likelihood that the document is a distracting non-relevant document; such a score can then be used to adjust the positive relevance score between the original query and the corresponding document.

Intuitively, a document with higher relevance score to the negative query representation can be assumed to be less relevant, thus the final score of this document can be computed as

$$S_{combined}(Q,D) = S(Q,D) - \beta \times S(Q_{neg},D) \qquad (1)$$

where Q_{neg} is a negative query representation and β is a parameter to control the influence of negative feedback. When $\beta = 0$, we do not perform negative feedback, and the ranking would be the same as the original ranking according to query Q. A larger value of β causes more penalization of documents similar to the negative query representation.

Equation (1) shows that *either* a high score of $S(Q,D)$ *or* a low score of $S(Q_{neg},D)$ would result in a high score of $S_{combined}(Q,D)$. This means that the proposed score combination may favor non-relevant documents if they have lower similarity to the negative model; this is risky because the negative query representation is only reliable for filtering out highly similar documents. Thus a more reasonable approach would be to only penalize documents which are most similar to the negative query model and avoid affecting the relevance scores of other documents. To achieve this goal, instead of penalizing all the documents in $\mathcal{U}$, we need to penalize only a subset of documents that are most similar to the negative query. We propose to use the following two heuristics to select documents for penalization (i.e., adjusting their scores using Equation (1)):

Heuristic 1 (Local Neighborhood): Rank all the documents in $\mathcal{U}$ by the negative query and penalize the top ρ documents.

Heuristic 2 (Global Neighborhood): Rank all the documents in $\mathcal{C}$ by the negative query. Select, from the top ρ documents of this ranked list, those documents in $\mathcal{U}$ to penalize.

In both cases, ρ is a parameter to control the number of documents to be penalized and would be empirically set. The two heuristics essentially differ in how this ρ value affects the number of documents in $\mathcal{U}$ to be penalized. In Heuristic 1, the actual number of documents in $\mathcal{U}$ to be penalized is fixed, i.e., ρ, but in Heuristic 2, it is dynamic and could be smaller than ρ, because the top ρ documents most similar to the negative query are generally not all in the set of $\mathcal{U}$. If we are to set ρ to a constant for all queries, intuitively Heuristic 2 can be more robust than Heuristics 1, which is confirmed in our experiments.

How do we compute the negative query repsentation Q_neg and the score $S(Q_{neg},D)$? A simple strategy is to combine all the negative information from $\mathcal{N}$ to form a single negative query representation, which would be referred to as "Single Negative Model" (*SingleNeg*). However, unlike positive information, negative information might be quite diverse. Thus, it is more desirable to capture negative information with more than one negative query model. Formally, let Q^i_{neg}, where $1 \leq i \leq k$, be k negative query models, we may compute $S(Q_{neg},D)$ as follows:

$$S(Q_{neg},D) = F(\bigcup_{i=1}^{k}\{S(Q^i_{neg},D)\})$$

where F is an aggregation function to combine the set of k values. We call this method "Multiple Negative Models" (*MultiNeg*).

3.2.3 Summary

We have discussed two general strategies with some variations for negative feedback, which can be summarized as follows: (1) **SingleQuery**: query modification strategy; (2) **SingleNeg**: score combination with a single negative query model; (3) **MultiNeg**: score combination with multiple negative query models. For both *SingleNeg* and *MultiNeg* models, we can use either of the two heuristics proposed in the previous subsection to penalize documents selectively. In the next subsection, we discuss some specific ways of implementing these general strategies in both vector space models and language models:

3.3 Negative Feedback in Vector Space Model

In vector space models, documents and queries are represented as vectors in a high-dimensional space spanned by terms. The weight of a term w in document D can be computed in many different ways and typically a similar measure such as dot product is used to score documents[15].

In our experiments, we use the following BM25 weight [11]:

$$\frac{(k_1+1)\times c(w,D)}{k_1((1-b)+b\frac{|d|}{avdl})+c(w,D)} \times \log(\frac{|\mathcal{C}|+1}{df(w)}) \qquad (2)$$

where k_1 and b are parameters. The weight of a query term is set to the raw term frequency, i.e., $c(w,Q)$. We compute the relevance score using the dot product: $S(Q,D) = \overrightarrow{Q} \cdot \overrightarrow{D}$ where $\overrightarrow{D}$ and $\overrightarrow{Q}$ represent document vector and query vector, respectively.

3.3.1 SingleQuery Strategy

The Rocchio method [12] is a commonly used feedback method in vector space models. The idea is to update a query vector with both relevant and non-relevant documents. When only non-relevant documents $\mathcal{N}$ are available, the Rocchio method can be written as

$$\overrightarrow{Q}_{new} = \overrightarrow{Q} - \gamma \times \frac{1}{|\mathcal{N}|}\sum_{D\in\mathcal{N}}\overrightarrow{D}. \qquad (3)$$

This gives us an updated query vector $\overrightarrow{Q}_{new}$, which can be used to rerank documents in $\mathcal{U}$.

3.3.2 SingleNeg Strategy

SingleNeg adjusts the original relevance score of a document with a single negative query. We compute the negative query as the center of negative documents, i.e., $\overrightarrow{Q}_{neg} = \frac{1}{|\mathcal{N}|}\sum_{D\in\mathcal{N}}\overrightarrow{D}$. Using Equation (1), the combined score of a document D is

$$S_{combined}(Q,D) = \overrightarrow{Q} \cdot \overrightarrow{D} - \beta \times \overrightarrow{Q}_{neg} \cdot \overrightarrow{D}. \qquad (4)$$

It is straightforward to verify that Equation (3) and (4) are equivalent. However, SingleNeg has the advantage of allowing us to penalize negative documents selectively using either of the two heuristics presented earlier.

3.3.3 MultiNeg Strategy

MultiNeg adjusts the original relevance score of a document with multiple negative queries which can be obtained, e.g., through clustering. In our experiments, we take each negative document as a negative query and use max as our aggregation function. Intuitively, max allows us to penalize any document that is close to at least one negative document. Thus

$$S(Q_{neg},D) = \max(\bigcup_{Q'\in\mathcal{N}}\{\overrightarrow{Q'} \cdot \overrightarrow{D}\}).$$

This score is then combined with $S(Q, D)$ to rerank the documents in $\mathcal{U}$. Again, we have two variants of this method corresponding to applying the two heuristics discussed above.

3.4 Negative Feedback for Language Models

KL-divergence retrieval model [9] is one of the most effective retrieval models in the language modeling framework. The relevance score is computed based on the negative KL-divergence between query model θ_Q and document model θ_D

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

where V is the set of words in our vocabulary. The document model θ_D needs to be smoothed and an effective method is Dirichlet smoothing [23]: $p(w|\theta_D) = \frac{c(w,D)+\mu p(w|\mathcal{C})}{|D|+\mu}$ where $p(w|\mathcal{C})$ is the collection language model and μ is a smoothing parameter.

Unlike vector space models, it is not natural to directly modify a query model using negative information in language model since no term can have a negative probability. In our recent work [19], several methods have been proposed for negative feedback in the language model framework. We adopt the methods there and combine them with the two heuristics discussed earlier for document penalization.

3.4.1 SingleNeg Strategy

SingleNeg adjusts the original relevance score of a document with a single negative model. Let θ_Q be the estimated query model for query Q and θ_D be the estimated document model for document D. Let θ_N be a negative topic model estimated based on the negative feedback documents $\mathcal{N} = \{L_1, ..., L_f\}$. In SingleNeg method, the new score of document D is computed as

$$S(Q, D) = -D(\theta_Q||\theta_D) + \beta \cdot D(\theta_N||\theta_D). \tag{5}$$

Note that we only penalize documents selected by either of the two heuristics.

We now discuss how to estimate negative model θ_N given a set of non-relevant documents $\mathcal{N} = \{L_1, ..., L_f\}$. We use the same estimation method as discussed in [19]. In particular, we assume that all non-relevant documents are generated from a mixture of a unigram language model θ_N (to generate non-relevant information) and a background language model (to generate common words). Thus, the log-likelihood of the sample $\mathcal{N}$ 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 is a mixture parameter which controls the weight of the background model and the background model is estimated with $p(w|\mathcal{C}) = \frac{c(w,\mathcal{C})}{\sum_w c(w,\mathcal{C})}$. Given a fixed λ ($\lambda = 0.9$ in our experiments), a standard EM algorithm can then be used to estimate parameters $p(w|\theta_N)$. The result of the EM algorithm gives a discriminative negative model θ_N which eliminates background noise.

3.4.2 SingleQuery Strategy

SingleQuery method is to update original query with negative information. Since every term has a non-negative probablity in a query model, there is no natural way to update original queries with negative information. However, given Equation (5), a SingleQuery method can be derived after applying algebra transformation and ignoring constants that do not affect document ranking in the following way

$$\begin{aligned} S(Q, D) &= -D(\theta_Q||\theta_D) + \beta \cdot D(\theta_N||\theta_D) \\ &\overset{\text{rank}}{=} \textstyle\sum_{w \in V} [p(w|\theta_Q) - \beta \cdot p(w|\theta_N)] \log p(w|\theta_D) \end{aligned}$$

The above equation shows that the weight of term w is $[p(w|\theta_Q) - \beta \cdot p(w|\theta_N)] \log p(w|\theta_D)$, which penalizes a term that has high probability in the negative topic model θ_N. In this way, $[p(w|\theta_Q) - \beta \cdot p(w|\theta_N)]$ can be regarded as the updated query model, which in some sense is the language modeling version of Rocchio. For consistence with vector space model, we use γ to replace β and use $[p(w|\theta_Q) - \gamma \cdot p(w|\theta_N)]$ as the updated query model. For this query model, we use the equation above to rerank all the documents in $\mathcal{U}$. Note that for SingleQuery method, we can not apply the two penalization heuristics.

3.4.3 MultiNeg Strategy

MultiNeg adjusts the original relevance scores with multiple negative models. We use the same EM algorithm as SingleNeg to estimate a negative model θ_i for each individual negative document L_i in $\mathcal{N}$. We then obtain f negative models and combine them as

$$\begin{aligned} S_{combined}(Q, D) &= S(Q, D) - \beta \cdot S(Q_{neg}, D) \\ &= S(Q, D) - \beta \cdot \max(\textstyle\bigcup_{i=1}^{f} \{S(Q_{neg}^{i}, D)\}) \\ &= -D(\theta_Q||\theta_D) + \beta \cdot \min(\textstyle\bigcup_{i=1}^{f} \{D(\theta_i||\theta_D)\}). \end{aligned}$$

4. CREATE TEST COLLECTIONS WITH SAMPLING

In order to evaluate the effectiveness of negative feedback methods, it is necessary to have test collections with sufficient difficult topics. However, TREC collections do not have many naturally difficult queries. In this section, we describe two sampling strategies to construct simulated test collections by converting easy topics to difficult topics.

In our problem formulation, a query is considered to be difficult if none of the top 10 documents retrieved by a retrieval model is relevant. Thus, in order to convert an easy query to a difficult one, our main idea of sampling methods is to delete some relevant documents of an easy query and assume these documents do not exist in the collection so that all top 10 documents are non-relevant. We now discuss two different ways to delete relevant documents:

Minimum Deletion Method: Given a query and a ranked document list for the query, we keep deleting the top ranked relevant document until none of the top 10 ranked documents of the list is relevant. We assume that the deleted relevant documents do not exist in the collection.

Random Deletion Method: Given a query and all of its relevant documents, we randomly delete a relevant document each time until none of the top 10 documents of the ranked list is relevant. Again, we assume that the deleted documents do not exist in the collection.

In both methods, we keep deleting relevant documents until none of top 10 ranked documents is relevant. Note that the constructed collections are dependent on retrieval models. After deletion, we obtain a new ranked list whose top 10 documents are irrelevant for a query. We then use these 10 irrelevant documents for negative feedback to rerank the next 1000 documents in this new ranked list.

5. EXPERIMENTS

To evaluate the effectiveness of negative feedback techniques, we construct our test collections based on two representative TREC data sets: ROBUST track and Web track data sets.

5.1 Data Sets

Our first data set is from the ROBUST track of TREC 2004. It has about 528,000 news articles [17]. On average, each document has 467 terms. We use all the 249 queries as our base query set. This data set is denoted by "ROBUST."

ROBUST		GOV	
LM	VSM	LM	VSM
$\mu = 2000$	$k_1 = 1.0, b = 0.3$	$\mu = 100$	$k_1 = 4.2, b = 0.8$

Table 1: Optimal parameter values.

Query Sets	ROBUST		GOV	
	LM	VSM	LM	VSM
QS0: P@10=0	26	25	54	63
QS12: 0.1 $\leq$P@10$\leq$ 0.2	57	61	56	46
QS46: 0.4 $\leq$P@10$\leq$ 0.6	67	78	6	6
ALL	150	164	116	115

Table 2: The query sets used in our experiments.

The second data set is the GOV data set used in the Web track of TREC 2003 and 2004. It is about 18 GB in size and contains $1,247,753$ Web pages crawled from the ".gov" domain in 2002. On average, each document has 1,094 terms. In our experiment, we only use the content of the pages for retrieval. There are 3 types of queries used in Web track: homepage finding, named page finding, and topic distillation. We use the queries with topic distillation type in both Web track 2003 and 2004. In total, we have 125 queries in our base set (50 from Web track 2003 and 75 from Web track 2004). We denote this data set by "GOV."

For both data sets, preprocessing involves only stemming but without removing any stopword. Since our goal is to study difficult queries, we construct different types of query sets from our base sets as follows.

5.1.1 Naturally Difficult Queries

The first type of query set consists of those naturally difficult queries. In this paper, we say that a query is a naturally difficult query if its P@10=0, given a retrieval model.

For both language models (LM) and vector space models (VSM), we use their standard ranking functions to select their naturally difficult queries respectively. We first optimize the parameters of μ for LM and k_1 and b for the VSM using the base set of queries on each data set. The optimal parameters are shown in Table 1. All these parameters are fixed in all the following experiments. Using the optimal parameter setting, we then select those queries whose P@10=0 as our naturally difficult queries. The row of QS0 in Table 2 shows the number of queries in this type of query sets.

5.1.2 Simulated Difficult Queries

Since there are not many naturally difficult queries, we further used the two deletion-based sampling methods to construct simulated difficult queries from easy ones. In our experiments, we use two types of easy queries. The first type consists of those queries whose P@10 satisfy 0.1 $\leq$P@10$\leq$ 0.2 (QS12 in Table 2) and the second consists of those queries whose P@10 satisfy 0.4 $\leq$P@10$\leq$ 0.6 (QS46 in Table 2). Again, all these queries are selected for the two retrieval models on the two data sets respectively.

The last type of query sets is the ALL query sets which are the union of the three types of query sets. Table 2 gives a summary of all the query sets used in our experiments.

5.2 Retrieval Effectiveness

Our experiment setup follows Section 3 to rerank the next unseen 1000 documents. We use two sets of performance measures: (1) Mean Average Precision (MAP) and Geometric Mean Average Precision (GMAP), which serve as good measures of the overall ranking accuracy. (2) Mean Reciprocal Rank (MRR) and Precision at 10 documents (P@10), which reflect the utility for users who only read the very top ranked documents.

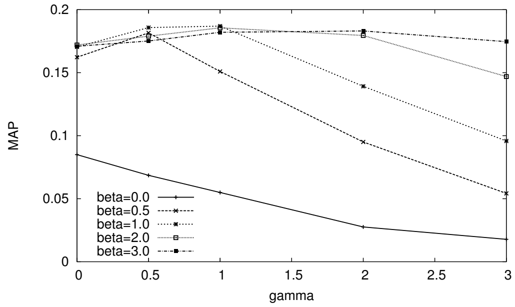

Figure 1: The performance of Rocchio feedback under different parameters.

5.2.1 Apply Existing Relevance Feedback

In this section, we use the Rocchio method in VSM to show that existing relevance feedback techniques do not work well if we only have negative information. The standard Rocchio method updates a query as

$$\overrightarrow{Q}_{new} = \alpha \times \overrightarrow{Q} + \beta \times \frac{1}{|\mathcal{R}|} \sum_{D \in \mathcal{R}} \overrightarrow{D} - \gamma \times \frac{1}{|\mathcal{N}|} \sum_{D \in \mathcal{N}} \overrightarrow{D}$$

where $\mathcal{R}$ is the set of relevant feedback documents. We use the query sets of "ALL" type. For any query, we first obtain its original ranking list. Starting from the top of the ranking list, we search downward until we arrive at a *cutting point*, before which we just find 10 irrelevant documents. All the documents before the cutting points, including both relevant and non-relevant documents, are used in the Rocchio feedback. The updated query vectors are then used to rerank the next 1000 documents starting from the cutting points. In our experiments, we set $\alpha = 1.0$ and vary β and γ. The results are shown in Figure 1. From this figure, we can see that if we have relevant information, i.e., $\beta > 0$, the MAP values can be improved dramatically. However, when we do not have any relevant information, i.e., $\beta = 0$, negative information always hurts MAP. This means that the existing relevance feedback techniques are not effective if only negative information is available for feedback, although they are very effective for positive feedback. This also shows that special techniques are needed for negative relevance feedback.

5.2.2 Overall Accuracy Comparison

Using naturally difficult query sets QS0, in this section, we study the effect of different negative feedback techniques. For both LM and VSM on the two data sets, we show their performance of the original ranking (OriginalRank) and the 5 negative feedback methods: SingleQuery means the SingleQuery strategy; SingleNeg1 and SingleNeg2 are the SingleNeg strategy plus Heuristic 1 and Heuristic 2 respectively; MultiNeg1 and MultiNeg2 are the MultiNeg strategy plus Heuristic 1 and Heuristic 2 respectively.

We vary the parameters for each method: γ from 0.01 to 1 for SingleQuery, β from 0.1 to 0.9 and ρ from 50 to 1000 for SingleNeg and MultiNeg methods. In Table 3, we compare the optimal performance (selected according to GMAP measure) of all methods. From this table, we have the following observations:

(1) LM approaches usually work better than VSM approaches. On the ROBUST data, LM can improve the MAP from 0.0293 to 0.0363, 23.8% relative improvement, but VSM can only improve from 0.0223 to 0.233, 4.4% relative improvement. On the GOV data, LM approaches can significantly improve over both OriginalRank and SingleQuery approaches, but VSM approaches can not consistently give improvements.

ROBUST+LM				
	MAP	GMAP	MRR	P@10
OriginalRank	0.0293	0.0137	0.1479	0.076
SingleQuery	0.0325	0.0141	0.2020	0.076
SingleNeg1	0.0325*	0.0147	0.2177	0.084
SingleNeg2	0.0330*+	0.0149	0.2130	**0.088**
MultiNeg1	0.0346*+	**0.0150**	**0.2368**	0.072
MultiNeg2	**0.0363***+	0.0148	0.2227	0.076

ROBUST+VSM				
	MAP	GMAP	MRR	P@10
OriginalRank	0.0223	0.0097	0.0744	**0.0416**
SingleQuery	0.0222	0.0097	0.0629	0.0375
SingleNeg1	0.0225*+	0.0097	0.0749	0.0375
SingleNeg2	0.0226*+	0.0097	0.0739	0.0375
MultiNeg1	0.0226*+	0.0099	0.0815	0.0375
MultiNeg2	**0.0233***+	**0.0100**	**0.0855**	**0.0416**

GOV+LM				
	MAP	GMAP	MRR	P@10
OriginalRank	0.0257	0.0054	0.0870	**0.0277**
SingleQuery	0.0297	0.0056	0.1070	**0.0277**
SingleNeg1	0.0300*	0.0056	0.1013	**0.0277**
SingleNeg2	0.0289*	0.0055	0.0899	0.0259
MultiNeg1	**0.0331***+	**0.0058**	**0.1150**	0.0259
MultiNeg2	0.0311*+	0.0057	0.1071	**0.0277**

GOV+VSM				
	MAP	GMAP	MRR	P@10
OriginalRank	0.0290	0.0035	0.0933	0.0206
SingleQuery	0.0301	**0.0038**	0.1085	0.0349
SingleNeg1	**0.0331***	**0.0038**	**0.1089**	**0.0396**
SingleNeg2	0.0298*	0.0036	0.0937	0.0349
MultiNeg1	0.0294	0.0036	0.0990	0.0349
MultiNeg2	0.0290	0.0036	0.0985	0.0333

Table 3: Optimal results of LM and VSM on the ROBUST and GOV data sets. * and + mean improvements over OriginalRank and SingleQuery, respectively, are statistically significant. We only show the significance tests on MAP values. Note that the values are not comparable across tables since each table corresponds to a different QS0 query set in Table 2.

	POS	NEG	MNEG
Group2 relevant	0.0039	0.0032	0.0081
Group2 irrelevant	0.0024	0.0034	0.0096

Table 4: Similarity between POS, NEG, MNEG learned from Group1 and relevant/irrelevant documents in Group2.

G37-07-3260432		G43-41-3966440	
VSM	LM	VSM	LM
xxxxx 22.15	mine 0.1166	pest 17.16	pest 0.1052
csic 20.85	industri 0.0475	mgmt 16.82	safeti 0.0861
quarri 20.13	miner 0.0357	4294 15.91	ds 0.0600
naic 19.48	metal 0.0319	ds 14.30	mgmt 0.0451
bitumin 18.65	or 0.0305	ipm 13.46	nih 0.0436

Table 5: Two examples of extracted negative models.

(2) For LM approaches, MultiNeg always works better than SingleQuery and SingleNeg. This shows that irrelevant documents may distract in different ways and do not form a coherent cluster. To verify this, we use the cutting point defined in Section 5.2.1 to form two groups of documents for each query: all documents before the cutting point form Group1 and the next 50 documents after the cutting point form Group2. We learn a positive (denoted as POS) and a negative language model (denoted as NEG) using the relevant and non-relevant documents in Group1. Using the exponential transform of negative KL-divergence as the similarity measure, we calculate the average similarity between POS/NEG and relevant/irrelevant documents of Group2. The average values over all queries are shown in Table 4. We can see that POS has a notably higher similarity to relevant documents than to irrelevant documents in Group2, but NEG does not have a notably higher similarity to irrelevant than relevant documents. In this table, we also show the results of multiple negative models (denoted as MNEG). Clearly, MNEG can distinguish between relevant and irrelevant documents better than NEG, confirming that negative documents are more diverse and MultiNeg is more appropriate for negative feedback.

(3) The results of VSM are mixed, and MultiNeg can not yield notable improvement on the GOV data. One possible reason is that the negative query vector generated using one single document in MultiNeg tends to over-emphasize rare terms due their high IDF values. In Table 5, we show two documents G37-07-3260432 and G43-41-3966440 in the GOV data set and their high-weight terms in the extracted negative query vectors. It is clear that VSM is biased towards those rare words such as “xxxxx” and “4294”, which makes the computed negative vectors less powerful to push down those similar irrelevant documents. For LM, the extracted terms are much better. This means that LM is more powerful to pick up more meaningful terms from negative documents and thus works better on GOV data.

This may also explain why SingleNeg in VSM is generally more effective than MultiNeg on the GOV data set: SingleNeg uses *multiple* negative documents to compute the negative models. While the rare words may bias the negative model computed from *a single* negative document in MultiNeg, their weights are small in SingleNeg since they are not common in all the negative documents.

5.2.3 Results with Simulated Difficult Queries

We have proposed two deletion-based sampling methods to make an easy query artificially difficult. In this section, we show the retrieval results on simulated difficult queries using ALL query sets. We only show the GMAP values in Table 6. For Random Deletion, we run it 10 times and the average performance values are reported here. In this table, we show the results of both deletion methods on both retrieval models and both data sets. Since Random Deletion deletes more relevant documents for each query than Minimum Deletion, it is expected that its overall performance is much lower than that of Minimum Deletion. The relative performance of different negative feedback methods is, however, similar to what we observed on the naturally difficult query sets, further confirming that the effectiveness of LM approaches and the MultiNeg strategy.

5.3 Effectiveness of Sampling Methods

5.3.1 Evaluation Methodology

To see whether a simulated test set generated using our sampling methods is as good as a test set with naturally difficult queries for evaluating negative feedback, we use both to rank different negative feedback methods based on their retrieval accuracy (e.g., MAP) and compare the two rankings; if they are highly correlated, it would indicate that the simulated test set can approximate a “natural” data set well.

Formally, assume that we have n negative retrieval functions. We can rank them based on their performance on the “gold standard” set (i.e., the naturally difficult queries). This would be our “gold standard ranking.” Similarly, we can use the simulated difficult queries to rank all these retrieval functions. We then compute the

	ROBUST+LM		ROBUST+VSM		GOV+LM		GOV+VSM	
	Minimum	Random	Minimum	Random	Minimum	Random	Minimum	Random
OriginalRank	0.0468	0.0126	0.0455	0.0115	0.0116	0.0069	0.0094	0.0056
SingleQuery	0.0467	0.0126	0.0454	0.0114	0.0119	0.0071	0.0104	0.0061
SingleNeg1	0.0473	0.0127	0.0451	0.0114	0.0118	0.0071	**0.0105**	**0.0063**
SingleNeg2	0.0475	0.0127	0.0454	0.0114	0.0120	0.0071	0.0103	0.0061
MultipleNeg1	0.0486	0.0129	0.0460	0.0116	**0.0129**	**0.0075**	0.0101	0.0059
MultipleNeg2	**0.0487**	**0.0130**	**0.0465**	**0.0117**	**0.0129**	0.0074	0.0100	0.0059

Table 6: The GMAP values of different methods on the simulated difficult query sets using Minimum and Random Deletion methods.

	ROBUST+LM		ROBUST+VSM		GOV+LM		GOV+VSM	
	MAP	GMAP	MAP	GMAP	MAP	GMAP	MAP	GMAP
Minimum	0.2990	0.4417	0.5015	0.7815	0.5382	0.7608	0.4932	0.6907
Random	0.3387	0.4537	0.3577	0.7417	0.5669	0.8071	0.5779	0.7271

Table 7: Kendall's τ coefficients between naturally difficult and simulated difficult queries.

correlation between these two ranking lists based on Kendall's τ rank coefficient. Given two ranking lists r_1 and r_2 of n retrieval functions, the coefficient is defined as

$$\tau(r_1, r_2) = \frac{2|\{(u, v) : r_1, r_2 \text{ agree on order of } (u, v), u \neq v\}|}{n \times (n-1)} - 1.$$

The range of the coefficient is between -1 and 1. When $\tau(r_1, r_2) > 0$, r_1 and r_2 are positively correlated. The larger the value, the higher the correlation. $\tau(r_1, r_2) = 1$ if r_1 and r_2 are exactly the same.

5.3.2 Results

We construct the simulated difficult queries using both Minimum and Random Deletion methods. Again we run the Random method 10 times and uses the average values to rank retrieval functions.

Our retrieval functions are from the 5 methods. For each method, we vary its parameter setting in a certain range. Each parameter setting will give us a different retrieval function. In total we have 110 retrieval functions.

Table 7 shows the Kendall's τ correlation coefficients between the naturally difficult queries QS0 and the simulated difficult queries on ALL query sets using the two deletion methods. From this table, we can see that both deletion methods are positively correlated with the naturally difficult queries. This confirm that our two deletion methods are reasonable to convert an easy query to a difficult one. Overall, Random Deletion is better than Minimum Deletion. Comparing two measures GMAP and MAP, we can see that the simulated difficult queries are more consistent with the naturally difficult queries on the GMAP measure. This indicates that GMAP is more appropriate as a measure on the simulated difficult queries than MAP. Indeed, GMAP has been used in ROBUST track to evaluate difficult queries and this shows the reasonableness of our deletion methods to simulate difficult queries.

5.4 Parameter Sensitivity Study

In this section, we study the parameter sensitivity. Due to space limit, we only show the results on ROBUST data set with the naturally difficult query set QS0; other results are similar.

Figure 2 shows the impact of γ on the SingleQuery method for both LM and VSM. We can see that SingleQuery can not effectively use the negative feedback information and it is quite sensitive if γ is larger. Figure 3 shows the impact of score combination parameter β where we set $\rho = 200$. All methods have the same level of sensitivities to β value. Figure 4 shows the impact of the penalization scope parameter ρ. It can be seen that SingleNeg1 and MultiNeg1 are very sensitive to this parameter, while SingleNeg2 and MultiNeg2 are more robust. These results confirm that Heuristic 2 is more stable than Heuristic 1 in general. Eventually, when ρ is large enough, the performance of SingleNeg2 and MultiNeg2 will drop as we penalize more documents which are not very similar to negative models. Finally, we study the impact of the number of feedback documents in MultiNeg. We set $f = 10$ but we only use a subset of these 10 documents in negative feedback. The result is in Figure 5 and it shows that we can get more improvement according to both MAP and GMAP if we use more documents in negative feedback. This means that our method can help more when a user accumulates more negative information.

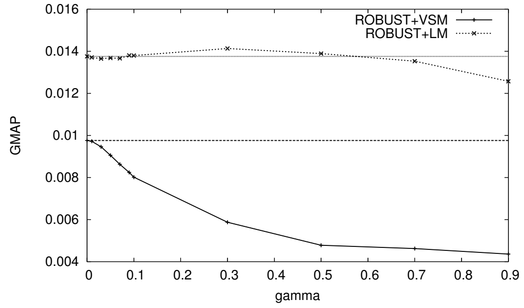

Figure 2: The impact of γ for SingleQuery method.

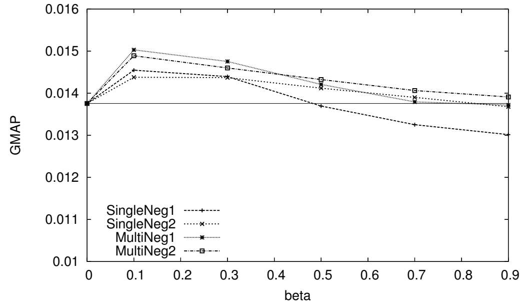

Figure 3: Impact of β for SingleNeg and MultiNeg.

6. CONCLUSIONS AND FUTURE WORK

Negative feedback is very important because it can help a user when search results are very poor. In this paper, we conducted a

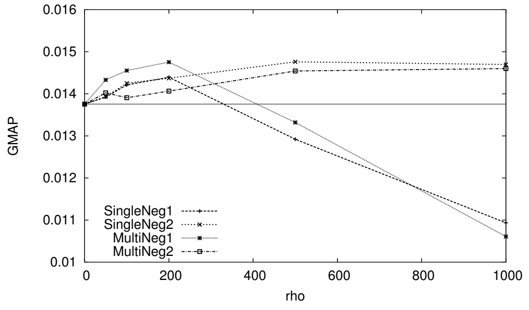

Figure 4: Impact of ρ in SingleNeg and MultiNeg.

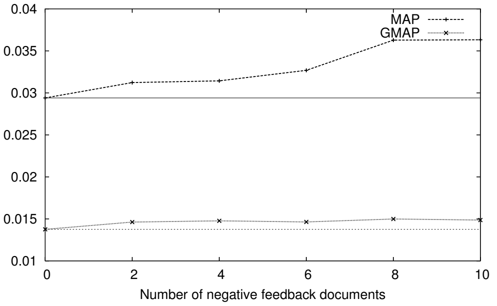

Figure 5: Impact of the number of feedback documents in MultiNeg.

systematic study of negative relevance feedback techniques. We proposed a set of general strategies for negative feedback and compared their instantiations in both vector space model and language modeling framework. We also proposed two heuristics to increase the robustness of using negative feedback information. Experiment results show that modeling multiple negative models is more effective than a single negative model and language model approaches are more effective than vector space model approaches. Studying negative feedback needs a test set with sufficient difficult queries. We further proposed two sampling methods to simulate difficult queries using easy ones. Our experiments show that both sampling methods are effective.

This work inspires several future directions. First, we can study a more principled way to model multiple negative models and use these multiple negative models to conduct constrained query expansion, for example, avoiding terms which are in negative models. Second, we are interested in a learning framework which can utilize both a little positive information (original queries) and a certain amount of negative information to learn a ranking function to help difficult queries. Third, queries are difficult due to different reasons. Identifying these reasons and customizing negative feedback strategies would be much worth studying.

7. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable suggestions. This work is in part supported by the National Science Foundation under award numbers IIS-0347933 and IIS-0713581.

8. REFERENCES

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

[2] C. Buckley. Why current IR engines fail. In *SIGIR'04*, pages 584–585, 2004.

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

[4] M. Dunlop. The effect of accessing non-matching documents on relevance feedback. *ACM TOIS*, 15(2), 1997.

[5] D. A. Evans and R. G. Lefferts. Design and evaluation of the CLARIT TREC-2 system. In D. Harman, editor, *TREC-2*, pages 137–150, 1994.

[6] D. Harman and C. Buckley. SIGIR 2004 Workshop: RIA and where can IR go from here. *SIGIR Forum*, 38(2):45–49, 2004.

[7] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, pages 76–84, 1996.

[8] M. Iwayama. Relevance feedback with a small number of relevance judgements: incremental relevance feedback vs. document clustering. In *SIGIR*, pages 10–16, 2000.

[9] J. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, pages 111–119, 2001.

[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] S. E. Robertson, S. Walker, S. Jones, M. M.Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, pages 109–126, 1995.

[12] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval SystemąłExperiments in Automatic Document Processing*, pages 313–323, 1971.

[13] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *JASIS*, 44(4):288–297, 1990.

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

[15] A. Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43, 2001.

[16] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. In *Proceedings of ACM SIGIR 1997*.

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

[18] E. M. Voorhees. Overview of the trec 2004 robust retrieval track. In *TREC2004*, 2005.

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

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

[21] 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. In *SIGIR*, pages 512–519, 2005.

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

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