Mining Specific and General Features in Both Positive and Negative Relevance Feedback

QUT E-Discovery Lab at the TREC'10 Relevance Feedback Track

Abdulmohsen Algarni, Yuefeng Li, Xiaohui Tao,
Computer Science Discipline, Queensland University of Technology, Australia
$\{$a1.algarni,y2.li, x.tao$\}$@qut.edu.au

Abstract

User relevance feedback is usually utilized by Web systems to interpret user information needs and retrieve effective results for users. However, how to discover useful knowledge in user relevance feedback and how to wisely use the discovered knowledge are two critical problems. However, understanding what makes an individual document good or bad for feedback can lead to the solution of the previous problem. In TREC 2010, we participated in the Relevance Feedback Track and experimented two models for extracting pseudo-relevance feedback to improve the ranking of retrieved documents. The first one, the main run, was a pattern-based model, whereas the second one, the optional run, was a term-based model. The two models consisted of two stages: one using relevance feedback provided by TREC'10 to expand queries to extract pseudo-relevance feedback; one using pseudo-relevance feedback to find useful patterns and terms according to their relevance and irrelevance judgements to rank documents. In this paper, the detailed description of those models is presented.

1 Introduction

Web users' personal interests and preferences can be drawn in their user profiles. In Web information gathering, user profiles are used by many works to search information for users according to their personal needs [2, 8]. To acquire user profiles, some techniques explicitly interview users [16]; some use user relevance feedback [21]. These mechanisms require user-effort in the user profile acquisition process. Attempting to release such burden from users, alternatively some automatic techniques have been developed to acquire user profiles from a collection of user personal information, for example, browsing history [2, 25]. User profiles acquired by such techniques, however, usually contain noise and uncertainties. Hence, a method to acquire user profiles effectively and efficiently (without the burden of user-effort) is an urgent need for personalized Web information gathering.

Relevance feedback has been used widely in the area of information retrieval. It has been reported effective when applying to different kinds of retrieval models [7, 19, 20, 22, 33]. The idea of relevance feedback is to involve the user in the retrieval process in order to improve the final result set. Some retrieval models also used pseudo relevance feedback [13, 14] especially when there were no relevance judgements available. In such models a small number of top-ranked documents in the initial retrieval results are assumed relevant, and then relevance feedback is applied [6].

The popular term-based IR models include the Rocchio algorithm [4,19], probabilistic models, Okapi BM25 [5,18], and language models including model-based methods and relevance models [7, 15, 17, 31, 33]. Generally speaking, in the vector space model, terms have been extracted from feedback by using the Rocchio algorithm. Those terms are used to form a new query vector by maximizing its similarity to relevant documents and minimizing its similarity to non-relevant documents [19]. In the language modelling approaches, the key elements are the probabilities of word sequences including both words and phrases (or sentences). They are often approximated by *n-gram* models [27], such as Unigram, Bigram or Trigram, for considering term dependencies.

IR models are the basis of ranking algorithm used in search engines to rank documents according to their relevance to a given query [3, 32]. Over the years, pattern-based approaches have been expected to outperform term-based techniques when discovering relevance features. Patterns are more discriminative and carry more "semantics". However, according to information retrieval (IR) experiments, few significant improvements have been achieved by using pattern-based methods to replace term-based methods [23,24]. When utilizing pattern mining techniques, people encountered two problems: (i) highly frequent patterns are usually general, whereas specific patterns are usually with low frequency (this is because the measuring methods for pattern learning, such as "support" and "confidences", appeared unsuitable in the filtering stage [9]); (ii) negative user feedback is difficult to use when revising the features extracted from positive user feedback. Relevance feature discovery is challenging [8].

A promising model, Relevance Feature Discovery (RFD), has been proposed by [10] for information filtering (IF) within the data mining community. The model has shown encouraging improvements of IF effectiveness. Closed sequential patterns were discovered in positive text documents, where a pattern was a set of terms that frequently appeared in a paragraph. The deployed method was applied to the extracted patterns to overcome the low frequency problem. Based on the positive features some negative documents were selected (called offenders) that were closed to the extracted features in the positive documents. The features were extracted from selected negative documents used for groups. Low-level features (terms) were devised based on both their appearances in the higher-level features (patterns) and their categories [1, 10]. The objective of relevance feature discovery is to find useful features available in a training set, including both positive and negative documents, to describe what users want. This is a particularly challenging task in modern information analysis, from both an empirical and a theoretical perspective [8, 11]. Motivated by these challenges, we proposed a relevance feature discovery model, and tested the model in the Relevance Feedback Track in TREC 2010. This Relevance Feedback track was designed to to investigate what makes an individual document good or bad for feedback.

The remainder of this paper is organized as follows. Section 2 presents an overview of the Relevance Feedback Track in TREC'10. Section 3 reviews the concepts of Rocchio and cosine similarity. The query expansion based on TREC provided feedback and Rocchio model are discussed in Section 4. Section 5 reviews the concept of patterns in text documents and a detailed reviews of Relevance Feature Discovery (RFD) model. Section 6 describes the optional run based on Rocchio model and pseudo-feedback. After that, the final retrieved results are discussed in Section 7, and the last section makes conclusions.

2 The Model

In response to a query, the first stage is to automatically retrieve a list of documents from the ClubWeb09 Category-B and rank them based on their similarity to the query. The key issue here is how to acquire user interest from limited information. In TREC'10 most of the queries have a limited number of terms, whereas the ClubWeb09 Category-B dataset has a large number of documents. Expanding the query at this stage without any feedback from the user could misleading. For example, there is many versions of interpretation that can be learned from a keyword "toilet" (e.g, toilet paper , toilet design, toilet suites, caroma toilet,...etc.). However, it is difficult to determine which one reflects the user's information need.

Based on that observation, we developed a model to work as follows:

1. Given a topic, 15000 relevant documents were extracted using Rocchio and cosine similarity via content search. The top 2500 documents were submitted as the base run results;

2. Using the highly frequent terms extracted from the user feedback (provided by TREC'10) to expand queries using Rocchio. The 15000 documents were re-ranked again using cosine similarity method;

3. The top 10 documents were selected as the positive feedback and the bottom 10 as negative. These pseudo-relevance feedback were used to update user profiles in two different runs.

    - **Main run:** The pseudo-relevance feedback went into the RFD model to generate three feature sets (the positive, specific terms, general terms, and negative, specific terms). The three feature sets were used to re-index the 15000 documents.
    - **Optional run:** The Rocchio was used to build user profiles from pseudo-relevance feedback; then used the feedback to re-index the 15000 documents.

4. The top 2500 ranked documents were submitted as the final results.

3 Rocchio and Cosine Similarity

In the vector space model all queries and documents are represented as vectors in $|V|$-dimensional space, where$V$ is the set of all distinct terms in the collection. Restricted by a fixed similarity metric, documents with similar content have similar vectors. However, the similarity of a document d to a query q is measured based on the terminological overlap between the query and the document. Thus, those relatively rare terms have a comparatively high weight. The documents are ranked by the magnitude of the angle between the document vector and the query vector.

The core of vector space model is $cosinemeasure$ that measures the angle between two vectors. The cosine between two vectors is determined as the dot product between each document vector $\overrightarrow{d}$ and the query vector $\overrightarrow{q}$, normalized by the lengths of the document and the query. The main function of cosine similarity can be generalized as follow:

$$cosine(q, d) = \frac{\overrightarrow{q}.\overrightarrow{d}}{|\overrightarrow{q}||\overrightarrow{d}|}$$

The vector space model does not specify how to set the document term weight and the query term weight, but in practice these weights are often calculated using their collection frequency and within-document frequency:

$$w_{q,t} = ln(1 + \frac{N}{f_t}) \quad (1)$$

$$w_{d,t} = 1 + ln(f_{d,t}) \quad (2)$$

In vector space model each document d is represented as a vector $\overrightarrow{d} = (d, ;d_n)$. According to a fixed similarity metric, documents with similar content have similar vectors. Each element d_n is represented by a set for terms $T = \{t_0, t_1, ;t_i\}$. The termset T of a document d is calculated as a combination of the statistics $TF(t_i; d)$ and $DF(t_i)$. The $termfrequency$ $TF(t_i; d)$ is the number of terms t_i occurred in the document d; the $documentfrequency$ $IDF(t_i)$ is the number of documents with term t_i occurred at least once [4].

Since the query length is constant for the evaluation of a single query, we can ignore this factor, while preserving ranking order. Bringing Equations 1 and 2 into the cosine measure, we have:

$$cosine(q, d)^{rank} = \frac{\sum\limits_{t \in q \cap d} (ln(1 + \frac{N}{f_t}) \times (1 + ln(f_{d,t})))}{\sqrt{\sum\limits_{t \in d} (1 + ln(f_{d,t}))^2}} \quad (3)$$

Similarity score have been calculated for all documents in ClubWeb09 dataset Category-B against queries using Eq. 3. The weight of each term t in a query q is set to 1 after text pre-processing include steaming and stopword removal. As a result, Eq. 3 can be re-generalized as:

$$cosine(q, d)^{rank} = \frac{\sum\limits_{t \in q \cap d} (1 \times f_{d,t})}{|f_{d,t}|}$$

The final result of this step is a ranked list of all documents with their weights. The top 2500 documents were selected as a result of the base run; the top 15000 documents were used as an input of the next stage.

4 Query Expansion

The main goal of query expansion is to optimize a query. A query is optimal if it ranks all relevant documents on top of those non-relevant. A query could lead to a good ranking result if it contains all features in relevant documents and disregards all features in non-relevant documents. However, the reformulation of an optimal query is difficult. The Rocchio algorithm [19] has been widely adopted when using relevant documents (D^+) and non-relevant documents (D^-) to reformulate an initial query q:

Table 1. A set of paragraphs

$Paragraph$	$Terms$
dp_1	t_1 t_2
dp_2	t_3 t_4 t_6
dp_3	t_3 t_4 t_5 t_6
dp_4	t_3 t_4 t_5 t_6
dp_5	t_1 t_2 t_6 t_7
dp_6	t_1 t_2 t_6 t_7

$$\overrightarrow{q} = \gamma \times \overrightarrow{q} + \alpha \frac{1}{|D^+|} \sum_{\overrightarrow{d} \in D^+} \frac{\overrightarrow{d}}{||\overrightarrow{d}||} - \beta \frac{1}{|D^-|} \sum_{\overrightarrow{d} \in D^-} \frac{\overrightarrow{d}}{||\overrightarrow{d}||} \quad (4)$$

where $\alpha = \beta = \gamma = 1.0$ in this presented work as suggested by [26].

4.1 Query Expansion with Given Feedback

Relevance feedback requires a user to classify documents into groups of relevant or non-relevant. Relevance feedback has been used widely in information retrieval. It has been reported effective in many IR models [7, 19, 20, 22, 33]. The feedback is used to expand queries and make better ranking of documents.

In TREC'10, with the main goal of understanding what makes an individual document good or bad for feedback, we used only one document explicitly provided by the user (NIST) to expand query $\overrightarrow{q}$. Thus, Eq. 4 can be simplified:

$$\overrightarrow{q} = \gamma \times \overrightarrow{q} + \alpha \sum_{\overrightarrow{d} \in D^+} \frac{\overrightarrow{d}}{||\overrightarrow{d}||}$$

The updated query $\overrightarrow{q}$ was used to $re-rank$ the 15000 documents retrieved in Section 3. The top 10 weighted documents were selected as pseudo positive feedback D^+; the lowest 10 ranked documents were selected as pseudo negative feedback D^-. They were used to expand the query again to $re-rank$ the documents. In TREC'10 we provided two runs: the main run using the Relevance Feature Discovery (RFD) model, and the optional run using the Rocchio model (see Eq. 4). More details are provided in the following sections.

Table 2. Frequent patterns and covering sets

$Frequent\ Pattern$	$Covering\ Set$
$\{\mathbf{t_3}, \mathbf{t_4}, \mathbf{t_6}\}$	$\{dp_2, dp_3, dp_4\}$
$\{t_3, t_4\}$	$\{dp_2, dp_3, dp_4\}$
$\{t_3, t_6\}$	$\{dp_2, dp_3, dp_4\}$
$\{t_4, t_6\}$	$\{dp_2, dp_3, dp_4\}$
$\{t_3\}$	$\{dp_2, dp_3, dp_4\}$
$\{t_4\}$	$\{dp_2, dp_3, dp_4\}$
$\{\mathbf{t_1}, \mathbf{t_2}\}$	$\{dp_1, dp_5, dp_6\}$
$\{t_1\}$	$\{dp_1, dp_5, dp_6\}$
$\{t_2\}$	$\{dp_1, dp_5, dp_6\}$
$\{\mathbf{t_6}\}$	$\{dp_2, dp_3, dp_4, dp_5, dp_6\}$

5 Relevance Feature Discovery (RFD) (main run)

5.1 Frequent and Closed Sequential Patterns

Table 1 lists a set of paragraphs for a sample document d, where $PS(d) = \{dp_1, dp_2, \ldots, dp_6\}$ with duplicate terms removed. Assume $min\_sup = 3$, ten frequent patterns would be extracted as shown in Table 2.

For a given topic, the RFD model extracts from a document set a set of features, including patterns and terms, and assigns them weights. The document set, usually called a training set and denoted as D, consists of a set of positive documents (D^+) and a set of negative documents (D^-). When splitting a document into paragraphs, a document d can also be represented by a set of paragraphs $PS(d)$.

Let $T = \{t_1, t_2, \ldots, t_m\}$ be a set of terms extracted from D^+; X be a set of terms (called a *termset*) in document d. $coverset(X)$ denotes the covering set of X for d, which includes all paragraphs $dp \in PS(d)$ where $X \subseteq dp$, i.e., $coverset(X) = \{dp | dp \in PS(d), X \subseteq dp\}$. The *absolute support* of X is the number of occurrences of X in $PS(d)$: $sup_a(X) = |coverset(X)|$. The *relative support* of X is the fraction of the paragraphs that contain the pattern: $sup_r(X) = \frac{|coverset(X)|}{|PS(d)|}$. A termset X is then called a *frequent pattern* if its sup_a (or sup_r) $\geq min\_sup$, a minimum support.

Given a set of paragraphs $Y \subseteq PS(d)$, we can also define its *termset*, which satisfies

$$termset(Y) = \{t | \forall dp \in Y \Rightarrow t \in dp\}.$$

By defining the closure of X as:

$$Cls(X) = termset(coverset(X))$$

a pattern (or termset) X is *closed* if and only if $X = Cls(X)$.

Let X be a closed pattern. We have

$$sup_a(X_1) < sup_a(X) \qquad (5)$$

for all patterns $X_1 \supset X$.

A taxonomy can be constructed by using closed patterns with *is-a* (or *subset*) relations. Table 2 contains three closed patterns, $< t_3, t_4, t_6 >$, $< t_1, t_2 >$, and $< t_6 >$, within ten frequent patterns. After pruning the non-closed patterns, a pattern taxonomy PT can be constructed, like $PT = \{\langle t_3, t_4, t_6\rangle, \langle t_1, t_2\rangle, \langle t_6\rangle\}$ in Table 2 when considering $\langle t_6\rangle$ a subset of $\langle t_3, t_4, t_6\rangle$.

Small patterns (e.g. $\langle t_6\rangle$) in a taxonomy are usually general because they have more chance to be used frequently. Vice versa, large patterns (e.g. $\langle t_3, t_4, t_6\rangle$) are relatively specific because they usually have a low frequency.

A sequential pattern $s =< t_1, \ldots, t_r >$ ($t_i \in T$) is an ordered list of terms. Denoted by $s_1 \sqsubseteq s_2$, a sequence $s_1 =< x_1, \ldots, x_i >$ is a sub-sequence of $s_2 =< y_1, \ldots, y_j >$, iff $\exists j_1, \ldots, j_i$ such that $1 \leq j_1 < j_2 \ldots < j_i \leq j$ and $x_1 = y_{j_1}, x_2 = y_{j_2}, \ldots, x_i = y_{j_i}$. Given $s_1 \sqsubseteq s_2$, we call s_1 a sub-pattern of s_2, and s_2 a super-pattern of s_1. To simplify the explanation, we refer to sequential patterns as patterns.

As the same as those defined for normal patterns, we define the *absolute support* and *relative support* for a pattern (an ordered *termset*) X in d. We also denote the covering set of X as $coverset(X)$, which includes all paragraphs $ps \in PS(d)$ such that $X \sqsubseteq ps$, i.e., $coverset(X) = \{ps | ps \in PS(d), X \sqsubseteq ps\}$. X is then called a *frequent pattern* if $sup_r(X) \geq min\_sup$. By using Eq. (5), a frequent sequential pattern X is *closed* if $\nexists$ any super-pattern X_1 of X such that $sup_a(X_1) = sup_a(X)$.

5.2 Deploying High-Level Patterns on Low-Level Terms

To overcome the problem of patterns with low-frequency, a method has been developed to deploy high level patterns over low-level terms. The evaluation of term supports (weights) in this paper is different from that in term-based approaches. For a term-based approach, the value of a term is scaled based on its appearance in documents. In our method, the value of terms are scaled based on their appearance in discovered patterns.

To improve the efficiency of the pattern taxonomy mining (PTM), an algorithm, *SPMining*(D^+, min_sup), was introduced by [29] and further developed in [9, 30] to find closed sequential patterns from positive documents D^+. The *SPMining* algorithm used the well-known *Apriori* property to narrow down the searching space.

Let SP_1, SP_2, ..., SP_n be the sets of discovered closed sequential patterns for all documents $d_i \in D^+ (i =$

$1, \cdots, n$), where $n = |D^+|$. For a given term t, its *weight* in discovered patterns is assigned by:

$$w(t, D^+) = \sum_{i=1}^{n} \sum_{t \in p \subseteq SP_i} \frac{sup_r(p, d_i)}{|p|} \tag{6}$$

where $|p|$ is the number of terms in p.

With weights assigned to the terms in D^+, a function can be used to rank and judge the relevance of incoming documents:

$$rank(d) = \sum_{t \in T} w(t)\tau(t, d)$$

where $w(t) = w(t, D^+)$; and $\tau(t, d) = 1$ if $t \in d$, otherwise $\tau(t, d) = 0$.

5.3 Mining Negative Patterns for Revising Low-Level Features

In general, the concept of relevance is subjective. Normally people can describe the relevance of a topic (or document) in specificity or exhaustivity, where "specificity" describes the extent to which the topic focuses on what users want, and "exhaustivity" describes the extent to which the topic discusses what users want. It is easy for human being to do so. However, it is very difficult to use these concepts for interpreting relevance features in text documents. In this section, we first discuss how to use the concepts for understanding the different roles of the low-level feature terms for answering what users want. We also present the ideas for accurately weighting terms based on their specificity and distributions in the discovered higher level features. In addition, we describe algorithms for both the discovery of higher level features and the revision of weights of low-level terms.

5.3.1 Specificity of Low-Level Features

A term's specificity describes the extent of the term to which the topic focuses on what users want. It is very difficult to measure the specificity of terms because a term's specificity depends on users' perspectives for their information needs [28]. Basically, we can understand the specificity of terms based on their positions in a concept hierarchy. For example, terms are more general if they are in the upper part of the LCSH (Library of Congress Subject Headings) hierarchy; otherwise, they are more specific. However, in many cases, a term's specificity is measured based on what topics we are talking about. For example, "knowledge discovery" will be a general term in data mining community; however it may be a specific term when we talk about information technology.

In order to understood how terms are grouped into three groups (positive specific terms, general terms and negative specific terms) based on their appearances in a training set. Given a term $t \in T$, its $coverage^+$ is the set of positive documents that contain t, and its $coverage^-$ is the set of negative documents that contain t. We assume that terms frequently used in both positive documents and negative documents are general terms. Therefore, we want to classify terms that are more frequently used in the positive documents into the positive specific category; and the terms that are more frequently used in the negative documents into the negative specific category.

Based on the above analysis, we define the *specificity* of a given term t in the training set $D = D^+ \cup D^-$ as follows:

$$spe(t) = \frac{|coverage^+(t)| - |coverage^-(t)|}{n}$$

where $coverage^+(t) = \{d \in D^+ | t \in d\}$, $coverage^-(t) = \{d \in D^- | t \in d\}$, and $n = |D^+|$. $spe(t) > 0$ means that term t is used more frequently in positive documents than in negative documents.

We present the following classification rules for determining the general terms G, the positive specific terms T^+, and the negative specific terms T^-:

$$G = \{t \in T | \theta_1 \leq spe(t) \leq \theta_2\},$$

$$T^+ = \{t \in T | spe(t) > \theta_2\},\ and$$

$$T^- = \{t \in T | spe(t) < \theta_1\}.$$

where θ_2 is an experimental coefficient, the maximum bound of the specificity for the general terms, and θ_1 is also an experimental coefficient, the minimum bound of the specificity for the general terms. We assume that $\theta_2 > 0$ and $\theta_2 \geq \theta_1$. It is easy to verify that $G \cap T^+ \cap T^- = \emptyset$. Therefore, $\{G, T^+, T^-\}$ is a partition of all terms.

To describe relevance features for a given topic, normally we believe that specific terms are very useful for the topic in order to distinguish it from other topics. However, many experiments show that using only specific terms is not good enough to improve the performance of RFD because user information needs cannot simply be covered by documents that contain only the specific terms. Therefore, the best way is to use the specific terms mixed with some of the general terms. We will discuss this issue in the evaluation section.

5.3.2 Revision of Discovered Features

In PTM, relevance features are discovered from a set of positive documents. To effectively use both higher level patterns and low-level terms, discovered patterns were deployed on the space of terms in order to evaluate the supports of terms obtained from the patterns.

Because of many noises in the discovered patterns (an inherent disadvantage of data mining), the evaluated supports are not accurate enough. To improve the effectiveness of PTM, in this paper, we use negative documents in the training set in order to remove the noises. Many people believed that negative documents can be helpful if they are used appropriately. The existing methods can be grouped into two approaches: revising terms that appear in both positive and negative documents; and observing how often terms appear in positive and negative documents. However, how much accuracy improvement can be achieved by using negative feedback still remains an open question.

There are two major issues for effectively using negative documents. The first one is how to select a suitable set of negative documents because we usually can obtain a very large set of negative samples. For example, a Google search can return millions of documents; however, only a few documents are interesting to a Web user. Obviously, it is not efficient to use all of negative documents. The second issue is how to revise the features discovered in the positive documents accurately.

In this research, we present an innovative solution for these issues. We firstly show how to select a set of negative samples. We also show the process of the revision.

If a document's rank (see Eq. (3)) is less than or equals to zero, this document is clearly negative to the system. If a negative document has a high rank, the document is called an offender [8] because it forces the system to make a mistake. The offenders are normally defined as the top-K negative documents in a ranked set of negative documents, D^-. The basics hypothesis is that the relevance features should be mainly discovered from the positive documents. Therefore, in our experiments, we set $K = \frac{n}{2}$, the half of the number of positive documents.

Once we select the top-K negative documents, the set of negative document D^- will be reduced to include only K offenders (negative documents). The next step is to classify terms into three categories, G, T^+, and T^-, based on D^+ and the updated D^-. We can easily verify that the experimental coefficients θ_1 and θ_2 satisfy the following properties if $K = \frac{n}{2}$:

$$0 \leq \theta_2 \leq 1, \;\; \texttt{and} \;\; -\frac{1}{2} \leq \theta_1 \leq \theta_2.$$

Now, we show the basic process of revising discovered features in a training set. This process can help readers to understand the proposed strategies for revising weights of low-level terms in different categories.

Formally, let DP^+ be the union of all discovered closed sequential patterns in D^+, DP^- be the union of all discovered closed sequential patterns in D^- and T be the set of terms that appear in DP^+ or DP^-, where a closed sequential pattern of D^+ (or D^-) is called a *positive pattern* (or *negative pattern*).

It is obviously that $\exists d \in D^+$ such that $t \in d$ for all $t \in T^+$ since $spe(t) > \theta_2 \geq 0$ for all $t \in T^+$. Therefore, for each $t \in T^+$, it can obtain an initial weight by the deploying method on D^+ (using the higher level features, see Eq. (2)).

For the term in $(T^- \cup G)$, there are two cases. If $\exists d \in D^+$ such that $t \in d$, t will get its initial weight by using the deploying method on D^+; otherwise it will get a negative weight by using the deploying methods on D^-.

The initial weights of terms finally are revised according to the following principles: increment the weights of the positive specific terms, decline the weights of the negative specific terms, and do not update the weights of the general terms. The details are described as follows:

$$weight(t) = \begin{cases} w(t) + w(t) \times spe(t), & \text{if } t \in T^+ \\ w(t), & \text{if } t \in G \\ w(t) - |w(t) \times spe(t)|, & \text{if } t \in T^- \end{cases}$$

where w is the initial weight (or the support in Eq. (2)).

5.3.3 Mining and Revision Algorithms

The process of the revision firstly finds features in the positive documents in the training set, including higher level positive patterns and low-level terms. It then selects top-K negative samples (called offenders) in the training set according to the positive features. It also discovers negative patterns and terms from selected negative documents using the same pattern mining technique that we used for the feature discovery in positive documents. In addition, the process revises the initial features and obtains a revised weight function. To understand this process clearly, we divided this process into two algorithms: *HLFMining* and *NRevision*. The former finds higher level positive features, selects top-K negative samples, discovers higher level negative features, and composes the set of terms. The latter revise the term weigh function based on the higher level features and the specificity of the terms.

Algorithm *HLFMining* describes the details of higher level feature discovery. It takes a training set and a minimum support, min_sup. It firstly abstracts patterns, DP^+, and then terms, T, in the set of positive documents in step 2 and step 3. It also gives the initial weights (step 4 and 5) to all terms based on their supports in DP^+. After that, the algorithm ranks the negative documents in D^- (step 6 to step 8), and selects offenders in step 9. Negative patterns and terms are also discovered in step 10 and 11. At last, it gives the initial weights to the terms that only appear in the negative patterns (step 12 and 13), and updates the set of terms (step 14).

The algorithm calls twice algorithm $SPMining$: one for positive documents and one for offenders (a part of

```
*HLFMining*($D$)
Input: A training set, $D = D^+ \cup D^-$, $min\_sup$ and $K$;
Output: extracted features $< DP^+, DP^-, T >$,
        updated training set $\{D^+, D^-\}$,
        and an initial term weight function, $w$.

Method:
1: $n = |D^+|, m = |D^-|$;
2: $DP^+ =$ SPMining$(D^+, min\_sup)$;
3: $T = \{t | t \in p, p \in DP^+\}$;
4: foreach $t \in T$ do
5:     $w(t) = support(t, D^+)$;
6: foreach $d \in D^-$ do
7:     $rank(d) = \Sigma_{t \in d \cap T} w(t)$;
8: let $D^- = \{d_0, d_1, ..., d_m\}$ in descendent ranking order,
9: $D^- = \{d_i | d_i \in D^-, rank(d_i) > 0, i < K\}$;
10: $DP^- =$ SPMining$(D^-, min\_sup)$;
11: $T_0 = \{t \in p | p \in DP^-\}$; // all terms in negative patterns
12: foreach $t \in (T_0 - T)$ do
13:     $w(t) = -support(t, D^-)$;
14: $T = T \cup T_0$;
```

```
*NRevision*( )
Input: A updated training set, $\{D^+, D^-\}$;
       extracted features $< T, DP^+, DP^- >$;
       the initial term weight function $w$; and experimental
       parameters $\theta_1$ and $\theta_2$, $-\frac{1}{2} \le \theta_1 \le \theta_2, 0 \le \theta_2 \le 1$.
Output: A term weight function.

Method:
1: $G = \emptyset, T^+ = \emptyset, T^- = \emptyset, n = |D^+|$;
2: foreach $t \in T$ do { //term partition
3:     $spe(t) = \frac{|\{d | d \in D^+, t \in d\}| - |\{d | d \in D^-, t \in d\}|}{n}$;
4:     if $spe(t) > \theta_2$
5:         then $T^+ = T^+ \cup \{t\}$;
6:         else if $spe(t) < \theta_1$
7:             then $T^- = T^- \cup \{t\}$;
8:             else $G = G \cup \{t\}$; }
9: foreach $t \in T^+$ do
10:     $weight(t) = w(t) + w(t) * spe(t)$;
11: foreach $t \in T^-$ do
12:     $weight(t) = w(t) - |w(t) * spe(t)|$;
```

negative documents). It also takes times for calculating weights of terms that appear in the discovered patterns, and sorts the negative documents that takes $O(mlog^m)$, where $m = |D^-|$. To compared with the time complexity of algorithm $SPMining$, the time of calculating weights spent here can be ignored. Therefore, the time complexity of this algorithm is $O(mlog^m)$ plus the time complexity of $SPMining$.

For a given training set $D = \{D^+, D^-\}$, using *HLFMining(D)*, we can obtain the extracted features $< DP^+, DP^-, T >$, and an initial weight function w over T. Given the experimental parameters θ_1 and θ_2, algorithm *NRevision* describes the details of revising the weights of the terms based on their specificity and distributions in both positive and negative patterns.

Step 1 initializes the sets for general terms G, positive specific terms T^+, and negative specific terms T^-. From step 2 to step 8, the algorithm partitions the terms into three categories. It firstly calculates the specificity for all terms, and then determines positive specific terms, negative specific terms and general terms based on the classification rules defined in Section 5.3.1 5.1. At last, it updates the initial weights of terms using the function *weight* defined in Section 5.3.2.

The time complexity of *NRevision( )* is mainly decided by the process of the partition (step 2 to step 8), where the calculation of the specificity dominates the process. For each term t, it takes $O(|d| \times (n + |D^-|)) = O(|d| \times n)$ for evaluating $spe(t)$, where $|d|$ is the average size of the documents, $|D^-|$ is the number of offenders and $|D^-| \le n$. Therefore, The time complexity of this algorithm is $O(|T| \times |d| \times n)$.

6 User Profiles Using Rocchio and Pseudo Relevance Feedback (Optional Run)

In order to compare with the pattern-based RFD model, a term-based model was implemented and tested for the optional run. The optional run used the Rocchio method to expand the query and re-rank the documents, as discuses in section 4 but using different pseudo relevance feedback. For each topic, we chose 150 terms in the pseudo positive documents based on *tf*idf* values. The test pre-processing again included the tasks of stopword removal and word stemming [12].

7 Process Review

Due to the large number of documents in ClueWeb09 Category-B and the limited number of terms in queries, at the first step, for each topic we retrieved about 15,000 candidate documents based content search. The documents were ranked using the cosine similarity values for documents vectors and query vectors, as discussed in Section 3. The top 2500 of the retrieved documents (15,000) were selected to submit as the base run results.

Then, the second step was to build a user profile for each topic in each run using the query terms and the relevance feedback provided by TREC'10 (one documents at a time). The retrieved 15000 documents from the first step were reranked using the user profile (discussed in Section 4) to select pseudo-relevance feedback. Then the top 10 weighted documents were selected as positive feedback documents D^+; and the lowest 10 weight documents were selected as negative feedback documents D^-. Pseudo-relevance feedback was used for the main algorithms to update user profiles and perform the optional run.

In the main run (RFD model), the relevance features were discovered from both positive and negative pseudo relevance feedback, using a model introduced in section 5. These relevance features consisted of high-level pattern features and low-level term features. Based on the high-level features, the low-level features were classified into three groups: positive specific terms, general terms, and negative specific terms. When applying negative patterns to revise the discovered features, we increased the weight of positive specific terms but declined that of negative specific terms based on specificity scour. Finally, we filtered the candidates based on document contents using the features discovered from positive and negative feedback, as discussed previously. The 15,000 candidates were re-ranked by accumulating the $weight(t)$ of features (see Algorithm *NRevision()*) that occurred in document contents. After that, the top 2,500 documents were selected and submitted as the final retrieved results against the given topic.

For the optional run, the Rocchio model was used to acquire user profiles from pseudo relevance feedback, as described in Section 6. The candidate documents were re-ranked based on the user profiles and the top 2,500 weighted documents were submitted as the final results of the optional run.

8 Conclusion

Under the framework set up by TREC 2010 Relevance Feedback Track, the work presented in this paper investigated two models: (i) the pattern-based RFD model that extracts pseudo-relevance feedback to improve the ranking of retrieved documents (the main run); (ii) a term-based model that extracts pseudo-relevance feedback to improve the ranking of retrieved documents (the optional run). Both models consist of two stages: one using the relevance feedback provided by NIST for query expansion to extract pseudo-relevance feedback; one using the pseudo-relevance feedback (including positive and negative feedback) to update user profiles. The exploration of using both positive and negative feedback for information retrieval was innovative, and the results are promising. Because negative feedback information is easily obtained comparing with positive feedback in traditional means, this work has opened a new door to improving the effectiveness of information retrieval.

Acknowledgements

The work presented in this paper was partially supported by Grants DP0988007 from the Australian Research Council.

References

[1] A. Algarni, Y. Li, Y. Xu, and R. Y. Lau. An effective model of using negative relevance feedback for information filtering. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1605–1608, New York, NY, USA, 2009. ACM.

[2] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelligence and Agent Systems*, 1(3-4):219–234, 2003.

[3] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 115–122, New York, NY, USA, 2008. ACM.

[4] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[5] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 1. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[6] K. Y. Lanbo Zhang Yi Zhang, Jadiel de Arma. UCSC at Relevance Feedback Track. In *Proceedings of the 18th Text Retrieval Conference (TREC 2009)*, 2009.

[7] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127, New York, NY, USA, 2001. ACM.

[8] Y. Li and N. Zhong. Mining Ontology for Automatically Acquiring Web User Information Needs. *IEEE Transactions on Knowledge and Data Engineering*, 18(4):554–568, 2006.

[9] Y. Li, X. Zhou, P. Bruza, Y. Xu, and R. Y. Lau. A two-stage text mining model for information filtering. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1023–1032, New York, NY, USA, 2008. ACM.

[10] Y. Li, A. Algarni, and N. Zhong. Mining positive and negative patterns for relevance feature discovery. In *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 753–762, New York, NY, USA, 2010. ACM.

[11] X. Ling, Q. Mei, C. Zhai, and B. Schatz. Mining multi-faceted overviews of arbitrary topics in a text collection. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–505, New York, NY, USA, 2008. ACM.

[12] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer, January 2007.

[13] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 255–264, New York, NY, USA, 2009. ACM.

[14] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[15] D. Metzler and W. B. Croft. Latent concept expansion using markov random fields. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 311–318, New York, NY, USA, 2007. ACM.

[16] S. E. Middleton, N. R. Shadbolt, and D. C. D. Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004.

[17] J. M. Ponte. A language modeling approach to information retrieval. Master's thesis, Amherst, MA, USA, 1998.

[18] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.

[19] J. Rocchio. *Relevance feedback in information retrieval*, volume In The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice Hall, 1971.

[20] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. pages 143–160, 1988.

[21] S. E. Robertson and I. Soboroff. The TREC 2002 filtering track report. In *Text REtrieval Conference*, 2002.

[22] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. pages 355–364, 1997.

[23] S. Scott and S. Matwin. Feature engineering for text classification. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 379–388, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[24] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[25] A. Sieg, B. Mobasher, and R. Burke. Web search personalization with ontological user profiles. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 525–534, New York, NY, USA, 2007. ACM.

[26] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32, New York, NY, USA, 1997. ACM.

[27] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, New York, NY, USA, 1999. ACM.

[28] X. Tao and Y. Li and N. Zhong. A Personalized Ontology Model for Web Information Gathering. IEEE Transactions on Knowledge and Data Engineering, IEEE computer Society Digital Library. IEEE Computer Society, 24 Aug 2010, http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.145

[29] S.-T. Wu, Y. Li, Y. Xu, B. Pham, and C. P. Automatic pattern taxonomy exatraction for web mining. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*, pages 242–248, Beijing, China, 2004.

[30] S.-T. Wu, Y. Li, and Y. Xu. Deploying approaches for pattern refinement in text mining. In *Proceedings of the Sixth International Conference on Data Mining*, pages 1157–1161, 2006.

[31] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 219–226, New York, NY, USA, 2008. ACM.

[32] C. C. Yang. Search engines information retrieval in practice. *J. Am. Soc. Inf. Sci. Technol.*, 61(2):430–430, 2010.

[33] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410, New York, NY, USA, 2001. ACM.