Mining Positive and Negative Patterns for Relevance Feature Discovery

Yuefeng Li
Discipline of Computer Science
Queensland University of Technology
Brisbane, QLD 4001, Australia
y2.li@qut.edu.au

Abdulmohsen Algarni
Discipline of Computer Science
Queensland University of Technology
Brisbane, QLD 4001, Australia
a1.algarni@qut.edu.au

Ning Zhong
Department of Life Science and Informatics
Maebashi Institute of Technology
Maebashi City, Japan
zhong@maebashi-it.ac.jp

ABSTRACT

It is a big challenge to guarantee the quality of discovered relevance features in text documents for describing user preferences because of the large number of terms, patterns, and noise. Most existing popular text mining and classification methods have adopted term-based approaches. However, they have all suffered from the problems of polysemy and synonymy. Over the years, people have often held the hypothesis that pattern-based methods should perform better than term-based ones in describing user preferences, but many experiments do not support this hypothesis. The innovative technique presented in paper makes a breakthrough for this difficulty. This technique discovers both positive and negative patterns in text documents as higher level features in order to accurately weight low-level features (terms) based on their specificity and their distributions in the higher level features. Substantial experiments using this technique on Reuters Corpus Volume 1 and TREC topics show that the proposed approach significantly outperforms both the state-of-the-art term-based methods underpinned by Okapi BM25, Rocchio or Support Vector Machine and pattern based methods on precision, recall and F measures.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Relevance feedback

General Terms

Algorithms, Experimentation, Performance

Keywords

Text mining, Pattern mining, Relevance feature discovery

1. INTRODUCTION

The objective of relevance feature discovery is to find useful features available in a training set, including both positive and negative documents, for describing what users want. This is a particularly challenging task in modern information analysis, from both an empirical and a theoretical perspective [20, 22]. This problem is also of central interest in many web personalized applications, and has received attention from researchers in Data Mining, Web Intelligence and Information Retrieval (IR) communities.

IR has provided many effective term-based methods to solve this challenge [28, 24]. The advantages of term-based methods include efficient computational performance; as well as mature theories for term weighting. Phrases have been used in some IR models, as phrases are more discriminative and carry more "semantics" than words. Currently, many researchers illustrated that phrases are useful and crucial for query expansion in building good ranking functions [36, 24, 4, 39].

Naturally, phrases have inferior statistical properties to words, and there are a large number of redundant and noisy phrases among them [31, 32]. Therefore, it is challenging to find the useful phrases only for text mining and classification. From the perspective of data mining, some researchers thought sequential patterns could be an alternative of phrases [40, 12]. Like words, patterns enjoy good statistical properties. Also, data mining has developed some techniques (e.g., maximal patterns, closed patterns and master patterns) for removing redundant and noisy patterns [46, 47, 42].

To use the advantages of data mining, pattern taxonomy models (PTM) [41, 40, 21] have been proposed for using closed sequential patterns in text classification. These pattern mining based approaches have shown a certain extent improvement on the effectiveness. However, fewer significant improvements are made compared with term-based methods.

There are two challenging issues for introducing pattern mining techniques to find relevance features in both positive and negative documents. The first is the low-support problem. Given a topic, large patterns are more specific for the topic, but with low supports (or low frequency). If we decrease the minimum support, there are a lot of noisy patterns would be discovered. The second issue is the misinterpretation problem that means the measures used in pattern mining (e.g., "support" and "confidence") turn out to be not suitable in using discovered patterns for answering what users want. For example, a highly frequent pattern (normally a short pattern) is usually a general pattern because it can be frequently used in both positive and negative documents. The difficult problem hence is how to use discovered patterns to accurately evaluate the weights of useful features in the training set.

Over the years, IR has developed many mature techniques which demonstrated that terms were useful features in text documents. However, many terms with larger weights are general terms because they can be frequently used in both relevant and irrelevant information. For example, term "LIB" may have larger weight

KDD'10, July 25–28, 2010, Washington, DC, USA.

than "JDK" in a certain of data collection; but we believe that term "JDK" is more specific than term "LIB" for describing "Java Programming Language"; and term "LIB" is more general than term "JDK" because term "LIB" is also frequently used in C and C++. Therefore, it is better to consider both terms' distributions and their specificities when we use them for text mining and classification.

In order to make a breakthrough for the two challenging issues, this paper proposes an innovative approach to evaluate weights of low-level terms according to both their specificity and their distributions in the higher level features, where the higher level features include both positive and negative patterns. Based on the specificity of terms, low-level terms can be classified into three categories: positive specific terms, general terms, and negative specific terms. In this way, various ways can be used to revise terms in the diffident categories based on the distributions of terms in the discovered patterns. In the implementation, we increment weights of positive specific terms, but decline weights of negative specific terms when we use negative patterns to revise discovered relevance features. We also conduct substantial experiments on the latest data collections, Reuters Corpus Volume 1 (RCV1) and TREC filtering assessor topics, to evaluate the proposed approach. The experimental results strongly support the proposed approach.

The remainder of this paper is organized as follows. Section 2 introduces a detailed overview of the related works. Section 3 reviews the concepts of patterns in text documents. Section 4 introduces the equations for evaluating weights of low-level terms based on discovered higher level features. Section 5 presents a new revision method for updating low-level terms based on negative patterns. The empirical results and discussion reported in Section 6 are followed concluding remarks in the last section.

2. RELATED WORK

IR has played an important role for the development of Web search engines, and involved a range of tasks: ad hoc search, filtering, classification and question answers. Currently, there are some big research issues in IR and Web search [6], such as evaluation, information needs, effective ranking and relevance. Relevance is a fundamental concept of information retrieval, which is classified into topical relevance and user relevance. The former discusses a document's relevance to a given query; and the latter discusses a document's relevance to a user.

Many IR models have been developed for relevance. There are two major classes in IR history: global methods and local methods [44, 23], where *global* means using corpus-based information, and *local* means using sets of retrieved or relevant documents. The popular term-based IR models include the Rocchio algorithm [30, 13], Probabilistic models and Okapi BM25 [38, 14] (more details about Rocchio algorithm and BM25 can be found in Section 6.2), and language models, including model-based methods and relevance models [26, 15, 50, 24, 39]. In a language model, the key elements are the probabilities of word sequences which include both words and phrases (or sentences). They are often approximated by *n-gram* models [36], such as Unigram, Bigram or Trigram, for considering term dependencies.

IR models are the basis of ranking algorithm that is used in search engines to produce the ranked list of documents [6]. A ranking model sorts a set of documents according to their relevance to a give query [8]. For a given query, phrases were very effective and crucial in building good ranking functions with large collections [24, 39].

Different to the ranking task, classification is the task of assigning documents to predefined categories. A comprehensive review of text classification methods can be found in [32]. To date, many classification methods, such as Naive Bayes, Rocchio, kNN and SVM have been developed in IR [5, 23].

SVM is one of the main machine learning methods for text classification [18, 45]. It also performed better on Reuters data collections than kNN and Rocchio [48, 17]. The classification problems include the single labeled and multi-labelled problem. The most common solution [51] to the multiple labelled problem is to decompose it into some independence binary classifiers, where a binary one is assigned to one of two predefined categories (e.g., positive category or negative category).

Data mining techniques were applied to text mining and classification by extracting co-occurring terms as descriptive phrases (*n-grams*) from document collections [16, 1, 11]. However, the effectiveness of the text mining systems using phrases as text representation showed no significant improvement since as a lot of training data incoming for an individual category (or a query), phrases are less help. Recently a concept-based model that analyzes terms on the sentence and document levels was introduced in [34], and ontology mining methods [20, 37] also provided some thoughts for text classification. However, they usually relied upon their employed natural language processing techniques or user defined ontologies.

Pattern mining has been extensively studied in data mining communities for many years. A variety of efficient algorithms such as Apriori-like algorithms, PrefixSpan, FP-tree, SPADE, SLPMiner and GST [9, 46, 49, 10, 33] have been proposed. These research works have mainly focused on developing efficient mining algorithms for discovering patterns in databases. In the field of text mining, however, interpreting useful patterns remains an open problem.

Typically, text mining discusses associations between terms at a broad spectrum level, paying little heed to duplications of terms, and labeled information in the training set [20]. Usually, the existing data mining techniques return numerous discovered patterns (e.g., sets of terms) from a training set. Not surprisingly, among these patterns, there are many redundant patterns [42]. Nevertheless, the challenging issue is how to effectively deal with the large amount of discovered patterns and terms with a lot of noises.

In the presence of these setbacks, closed patterns used in data mining community have turned out to be an alternative to phrases [41, 12] because patterns enjoy good statistical properties like terms. To effectively use closed patterns in text classification, a deploying method in [19] has been proposed to compose all closed patterns of a category into a vector that included a set of terms and a term-weight distribution. The pattern deploying method has shown encouraging improvements on effectiveness in comparing with traditional IR models [40, 21]. The similar research was also published in [47] for developing a new methodology of post-processing of pattern mining, pattern summarization, which grouped patterns into some clusters and then composed patterns in the same cluster into a master pattern that consists of a set of terms and a term-weight distribution.

In regard to the aforementioned problem of redundancy and noises, it is still a challenging issue for pattern-based methods to deal with low frequency patterns. By way of illustration, a short pattern (normally a highly frequent pattern with large support) is usually a general pattern, or a large pattern (a low frequent pattern with small support) could be a specific one.

In summary, we can group the existing methods for finding relevance features into three approaches. The first approach is to revise feature terms that appear in both positive samples and negative samples (e.g., Rocchio-based models [25] and SVM [27, 7]). This heuristic is obvious when people assume that terms are isolated atoms. The second approach is based on how often terms appear or do not appear in positive documents and negative documents (e.g.,

probabilistic based models [43] or BM25 [28, 29]). The third one is to describe features as positive patterns [41, 40]. In this paper, we further develop the third one by effectively using both higher-lever features (positive and negative patterns) and low-level terms.

3. DEFINITIONS

Relevance feature discovery is to find useful features, including patterns, terms and their weights, in a training set D, which consists of a set of positive documents, D^+, and a set of negative documents, D^-. In this paper, we assume that all documents are split into paragraphs. So a given document d yields a set of paragraphs $PS(d)$. To clearly understand the concepts of patterns, we introduce normal patterns and closed patterns first, and then we discuss sequential closed patterns. These definitions can be found in [40] or [21].

3.1 Frequent and Closed Patterns

Let $T = \{t_1, t_2, \ldots, t_m\}$ be a set of terms which are extracted from D^+. Given a *termset* X, a set of terms, in document d, $coverset(X) = \{dp | dp \in PS(d), X \subseteq dp\}$. Its *absolute support*

$$sup_a(X) = |coverset(X)|;$$

and its *relative support*

$$sup_r(X) = \frac{|coverset(X)|}{|PS(d)|}.$$

A termset X is called *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 define its *termset*, which satisfies

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

Let $Cls(X) = termset(coverset(X))$ be the closure of X. We call X *closed* if and only if $X = Cls(X)$.

Let X be a closed pattern. We have

$$sup_a(X_1) < sup_a(X) \quad (1)$$

for all pattern $X_1 \supset X$.

Patterns can be structured into a taxonomy by using the *is-a* (or *subset*) relation and closed patterns. Put simply, a pattern taxonomy is described as a set of patterns, and the relation in the taxonomy is the subset relation.

Smaller patterns in the taxonomy are usually more general because they could be used frequently in both positive and negative documents; but larger patterns in the taxonomy are usually more specific since they may be used only in positive documents.

3.2 Closed Sequential Patterns

A sequential pattern $s =< t_1, \ldots, t_r > (t_i \in T)$ is an ordered list of terms. A sequence $s_1 =< x_1, \ldots, x_i >$ is a sub-sequence of another sequence $s_2 =< y_1, \ldots, y_j >$, denoted by $s_1 \sqsubseteq s_2$, 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 usually say s_1 is a sub-pattern of s_2, and s_2 is a super-pattern of s_1. In the following, we refer to sequential patterns as patterns.

Given a sequential pattern X in document d, $coverset(X)$ is still used to denote the covering set of X, which includes all paragraphs $ps \in PS(d)$ such that $X \sqsubseteq ps$, i.e., $coverset(X) = \{ps | ps \in PS(d), X \sqsubseteq ps\}$. Its *absolute support* and *relative support* are defined as the same as for the normal patterns.

A sequential pattern X is called a *frequent pattern* if its relative support $\geq min\_sup$, a minimum support. The property of closed patterns (see Eq. (1)) can be used to define closed sequential patterns. A frequent sequential pattern X is called *closed* if not $\exists$ any super-pattern X_1 of X such that $sup_a(X_1) = sup_a(X)$.

4. THE DEPLOYING METHOD

In this section, we develop equations for deploying higher level patterns over low-level terms by evaluating term supports based on their appearances in patterns. The evaluation of term supports (weights) in this paper is different from term-based approaches. For a term-based approach, the evaluation of a given term's weight is based on its appearances in documents. In this research, terms are weighted according to their appearances in discovered patterns.

To improve the efficiency of the pattern taxonomy mining (PTM), an algorithm, *SPMining*(D^+, min_sup) [41], was proposed (also used in [40, 21]) to find closed sequential patterns for all documents $\in D^+$, which used the well-known *Apriori* property in order to reduce the searching space. For all positive documents $d_i \in D^+$, the *SPMining* algorithm can discover all closed sequential patterns, SP_i, based on a given min_sup. (We do not repeat this algorithm here because of the length limitation of the paper.)

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 support (or called *weight*) in discovered patterns can be described as follows:

$$support(t, D^+) = \sum_{i=1}^{n} \frac{|\{p | p \in SP_i, t \in p\}|}{\sum_{p \in SP_i} |p|} \quad (2)$$

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

Table 1 illustrates an example of sets of discovered closed sequential patterns for $D^+ = \{d_1, d_2, \cdots, d_5\}$. For example, term $global$ appears in three documents (d_2, d_3 and d_5). Therefore, its support is evaluated based on patterns in the sets of closed sequential patterns that contain $global$:

$$support(global, D^+) = \frac{2}{4} + \frac{1}{3} + \frac{1}{3} = \frac{7}{6}.$$

Table 1: Example of Pattern Mining

Doc.	Discovered Closed Sequential Patterns (SP_i)
d_1	{⟨*carbon*⟩ , ⟨*carbon*, *emiss*⟩, ⟨*air*, *pollut*⟩ }
d_2	{⟨*greenhous*, *global*⟩, ⟨*emiss*, *global*⟩}
d_3	{⟨*greenhous*⟩, ⟨*global*, *emiss*⟩}
d_4	{⟨*carbon*⟩, ⟨*air*⟩, ⟨*air*, *antarct*⟩}
d_5	{⟨*emiss*, *global*, *pollut*⟩}

After the supports of terms have been computed from the training set, the following rank will be assigned to every incoming document d to decide its relevance:

$$rank(d) = \sum_{t \in T} weight(t)\tau(t, d) \quad (3)$$

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

5. 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.

5.1 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 [37]. 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 this paper, we discuss 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 relevance feature discovery 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.

5.2 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 [20] 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 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.

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: $T = T \cup T_0$;

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 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.

NRevision( )

Input: A updated training set, $\{D^+, D^-\}$;
extracted features $< T, DP^+, DP^- >$;
the initial term weight function w; and experimental parameters θ_1 and θ_2, $-\frac{1}{2} \leq \theta_1 \leq \theta_2, 0 \leq \theta_2 \leq 1$.
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: $weight(t) = w(t) - |w(t) * spe(t)|$;

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 5.1. At last, it updates the initial weights of terms using the function *weight* defined in Section 5.2.

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 $|D^-| \leq n$. Therefore, The time complexity of this algorithm is $O(|T| \times |d| \times n)$.

6. EVALUATION

The main hypothesis that has been proposed in this paper is that negative feedback documents should be useful to remove the noises

in discovered features in the positive documents. To support this hypothesis, this section discusses the testing environment including the data collection, baseline models, and evaluation methods. It also reports the results and the discussions for the following main points: the proposed model is significant comparing with the baseline models on the effectiveness; and the classification rules are useful to reduce the noises in the discovered features. In addition, it provides recommendations for offender selection and the use of specific terms and general terms for describing user information needs. The proposed model is a supervised approach that needs a training set including both positive documents and negative documents.

6.1 Data

Reuters Corpus Volume 1 (RCV1) is used to test the effectiveness of the proposed model. RCV1 corpus consists of all and only English language stories produced by Reuter's journalists between August 20, 1996, and August 19, 1997, a total of 806,791 documents that cover very large topics and information. TREC (2002) has developed and provided 50 assessor topics [28] for the filtering track, aiming at building a robust filtering system. These topics were developed by human assessors of the National Institute of Standards and Technology (NIST)(i.e., assessor topics). The relevance judgements on RCV1 have also been made by the assessors of NIST. The assessor topics are more reliable than any artificially constructed topics [35]. For each topic, some documents in RCV1 are divided into a training set and a testing set. According to Buckley and others [2], 50 topics are stable and enough for high quality experiments. This research uses RCV1 and the 50 assessor topics to evaluate the proposed model.

Documents in RCV1 are marked in XML. To avoid bias in experiments, all of the meta-data information in the collection have been ignored. The documents are treated as plain text documents by preprocessing the documents. The tasks of removing stop-words according to a given stop-words list and stemming terms by applying the Porter Stemming algorithm are conducted.

6.2 Baseline Models and Setting

We group baseline models into two categories. The first category includes the up-to-date pattern mining based methods (frequent patterns, frequent closed patterns, sequential patterns, and sequential closed patterns) and n-grams. The second category includes the well-known term-based methods: Rocchio, BM25 and SVM models.

In the experiments, we also divide our approach into two stages. In the first stage, we use only positive patterns in the training sets. The model, called PTM, discovers sequential closed patterns from positive documents, deploys discovered patterns on their terms using Eq. (2) and ranks documents using Eq. (3) as well. In the second stage, we use both positive and negative patterns to refine discovered features based on *HLFMining* and *NRevision* algorithms. The proposed model is called RFD (Relevance Feature Discovery model).

The concepts of all kinds of patterns discussed in this section are defined in Section 3. For the pattern-based models, we rank a document based on the total relative supports of discovered patterns that appear in the document. We also set $min\_sup = 0.2$ (relative support) for all models that use patterns. The n-gram model uses *3-grams*.

The Rocchio model used a Centroid to describe a topic as follows:

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

There are two recommendations for setting parameters α and β in the Rocchio model [3, 13]: $\alpha = 16$ and $\beta = 4$; and $\alpha = \beta = 1.0$. We have tested both recommendations on RCV1 and found $\alpha = \beta = 1.0$ was the best one. Therefore, we let $\alpha = \beta = 1.0$ in the above equation.

BM25 [28] is one of state-of-the-art term-based models. The term weights are estimated as follows:

$$W(t) = \frac{tf \cdot (k_1 + 1)}{k_1 \cdot ((1-b) + b\frac{DL}{AVDL}) + tf} \cdot \log \frac{\frac{(r+0.5)}{(n-r+0.5)}}{\frac{(R-r+0.5)}{(N-n-R+r+0.5)}}$$

where N is the total number of documents in the training set; R is the number of positive documents in the training set; n is the number of documents which contain term t; r is the number of positive documents which contain term t; tf is the term frequency; DL and *AVDL* are the document length and average document length, respectively; and k_1 and b are the experimental parameters (the values of k_1 and b are set as 1.2 and 0.75, respectively, in this paper).

SVM is inherently a two-class classifier, and it is a very effective text classifier. SVM was also used in [4] as a classifier for pseudo-relevance feedback, where multiple features, such as term distribution, co-occurrence with query terms, and term proximity. The linear SVM achieved the best performance on the Reuters-21578 data collection for document classification [48, 23].

Therefore, for the relevance feature feedback, we used the following decision function in SVM:

$$h(x) = sign(w \cdot x + b) = \begin{cases} +1 & \text{if } (w \cdot x + b) > 0 \\ -1 & \text{otherwise} \end{cases}$$

where x is the input object; $b \, \epsilon \, \Re$ is a threshold and $w = \sum_{i=1}^{l} y_i \alpha_i x_i$ for the given training data: $(x_i, y_i), ..., (x_l, y_l)$, where $x_i \, \varepsilon \, \Re^n$ and $y_i = +1(-1)$, if document x_i is labeled positive (negative). $\alpha_i \, \epsilon \, \Re$ is the weight of the sample x_i and satisfies the constraint:

$$\forall_i : \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \quad (5)$$

To compare with other SVM variants, the SVM here is used to rank documents rather than to make a classification or a binary decision, and it only uses term based features. For this purpose, threshold b can be ignored. For the documents in a training set, we know only what are positive (or negative), but not which one is more important. To avoid this bias, we assign the same α_i value (i.e., 1) to each positive document first, and then determine the same α_i (i.e., $\acute{\alpha}$) value to each negative document based on Eq. (5). Therefore, we use the following weighting function to estimate the similarity between a testing document and a given topic:

$$weight(d) = w \cdot d$$

where $\cdot$ means *inner product*; d is the term vector of the testing document; and

$$w = (\sum_{d_i \in D^+} d_i) + (\sum_{d_j \in D^-} d_j \acute{\alpha}).$$

For each topic, we also choose 150 terms in the positive documents, based on *tf*idf* values for all term-based baseline models.

Table 2: Comparison of all pattern (phrase) based methods on all topics.

Method	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
RFD	**0.557**	**0.472**	**0.493**	**0.470**	**0.513**
PTM Deploying	0.496	0.430	0.444	0.439	0.464
Seq Patterns	0.401	0.343	0.361	0.385	0.384
Seq Closed Ptns	0.406	0.353	0.364	0.390	0.392
Freq Patterns	0.412	0.352	0.361	0.386	0.384
Freq Closed Ptns	0.428	0.346	0.361	0.385	0.387
n-Gram	0.401	0.342	0.361	0.386	0.384
$\%chg$	**+12.30**	**+9.75**	**+11.18**	**+6.92**	**+10.44**

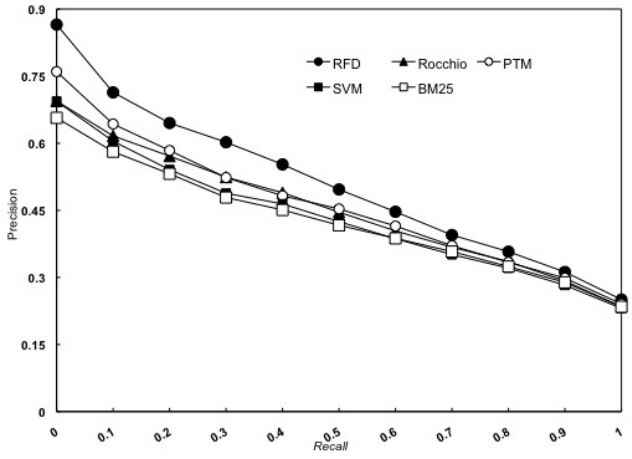

Figure 1: Comparison the results in all assessing topics.

6.3 Results

The effectiveness is measured by five different means: The average precision of the top 20 documents, F_1 measure, Mean Average Precision (MAP), the break-even point (b/p), and Interpolated Average Precision (IAP) on *11-points*. Precision (p), Recall(r) and F_1 are calculated by the following functions:

$$p = \frac{TP}{TP+FP}, r = \frac{TP}{TP+FN}, F_1 = \frac{2pr}{p+r};$$

where TP is the number of documents the system correctly identifies as positives; FP is the number of documents the system falsely identifies as positives; FN is the number of relevant documents the system fails to identify. The larger a *top-20*, MAP, *b/p*, IAP or F_1-measure score is, the better the system performs. *11-points* measure is also used to compare the performance of different systems by averaging precisions at 11 standard recall levels (i.e., recall=0.0, 0.1, ..., 1.0).

The statistical method, paired two-tailed *t-test*, is also used to analyze the experimental results. If the associated p-value is low ($<$0.05), that shows the difference in means across the paired observations is significant.

RFD is firstly compared with n-grams and other pattern-based models, especially PTM, the best one of the existing pattern-based models. RFD is also compared with the state-of-the-art term-based methods underpinned by Rocchio, BM25 and SVM for each variable $top-20$, b/p, MAP, IAP and $F_{\beta=1}$ over all the 50 assessing topics, respectively.

Table 3: Comparison results of all models in all assessing topics.

	top-20	*MAP*	$F_{\beta=1}$	*b/p*	*IAP*
RFD	**0.557**	**0.4932**	**0.4696**	**0.4724**	**0.5125**
Rocchio	0.474	0.4305	0.4299	0.4201	0.4523
SVM	0.453	0.4092	0.4211	0.4083	0.4353
BM25	0.445	0.4069	0.4140	0.4074	0.4281
$\%chg$	**+17.51%**	**+14.56%**	**+9.25%**	**+12.46%**	**+13.32%**

Table 4: p-values for all models comparing with RFD model in all assessing topics.

	top-20	*MAP*	$F_{\beta=1}$	*b/p*	*IAP*
PTM	0.01006	0.00025	0.00013	0.00262	0.00010
Rocchio	0.00302	0.00368	0.00341	0.00749	0.00344
SVM	0.00014	0.00005	0.00011	0.00085	0.00005
BM25	0.00032	0.00039	0.00031	0.00170	0.00019

6.3.1 RFD vs Pattern-Based Models and n-Grams

The results of overall comparisons between RFD, n-grams and other pattern-based models are presented in Table 2, where $\%chg$ means the percentage change of RFD over PTM. Noted earlier, pattern-based methods struggle in some topics as too much noise is generated in the discovery of positive patterns. The most important findings revealed in this table are that sequential closed patterns (Seq Closed Ptns) perform better than n-grams and other pattern-based models for the important measures (MAP and F_1), and that PTM Deploying largely outperforms sequential closed patterns.

The result also supports the superiority of using sequential closed patterns in text mining and highlights the importance of the adoption of proper pattern deploying methods on terms for using discovered patterns in text documents.

In order to see the effectiveness of using both positive and negative patterns for relevance feature discovery, we also compare RFD with PTM which uses positive patterns only. As shown in Table 2, RFD achieves excellent performance with 10.12% (max 12.30% and min 6.92%) in percentage change on average for all five measures.

6.3.2 RFD vs Term-Based Models

The proposed method RFD is also compared with term-based baseline models including Rocchio, BM25, and SVM. The experimental results on all 50 assessing topics are reported in Table 3 with the percentage changes comparing with the other best model's results.

As shown in Table 3, the proposed new model RFD has achieved the best performance results for the assessor topics. The average percentage of improvement over the standard measures is 13.42% with the maximum of 17.51% and minimum 9.25% comparing with the best result in Table 3.

In summary, the improvements are consistent and very significant on all five measures as shown by the results of *11-points* on all 50 assessing topics in Figure 1; and the *t-test* p values in Table 4, which indicate that the proposed model RFD is extremely statistically significant. Therefore, we conclude that the revision of low-level features based on their specificity and their distributions in both positive and negative patterns is an exciting achievement for the discovery of relevance features in text documents.

6.4 Discussion

The main process of RFD consists of two steps: offender selection and the revision of the weights of low-level terms that are classified into three categories. In this section, we discuss the is-

Table 5: Statistical information for RFD ($\theta_1 = 0.2$,$\theta_2 = 0.3$) with different values of K.

K	Average number of training documents			Average number of extracted terms			Average weight of extracted terms			*top-20*	*MAP*	$F_{\beta=1}$
	Positive	Negative	Offenders	$\#T^+$	$\#G$	$\#T^-$	$w(T^+)$	$w(G)$	$w(T^-)$			
$\|D^+\|/2$	12.78	41.3	6.54	23.54	22.36	231.78	4.159	1.400	-0.551	0.557	0.493	0.470
$\|D^+\|$	12.78	41.3	10.18	20.78	20.74	280.18	3.060	1.270	-2.964	0.537	0.463	0.450
$\|D^-\|$	12.78	41.3	39.92	14.20	15.28	539.36	1.858	0.890	-71.202	0.273	0.278	0.294

Table 6: Results of using different values for θ_1 and θ_2 in all assessing topics.

θ_1	θ_2	$\#T^+$	$\#G$	$\#T^-$	b/p	*top-20*	MAP	$F_{\beta=1}$	*IAP*
0.2	**0.3**	23.54	22.36	231.78	0.47243	0.55700	0.49315	0.46959	0.51253
-0.2	0.5	2.6	231.62	43.46	0.46727	0.55500	0.48951	0.46753	0.51059
-0.2	0.4	6.46	227.76	43.46	0.46934	0.55000	0.48954	0.46806	0.51089
0.0	0.4	6.46	139.92	131.3	0.46962	0.55500	0.48997	0.46813	0.51127
0.1	0.2	37.5	36.72	203.46	0.47340	0.55500	0.49238	0.46936	0.51194

Table 7: Statistical information RFD and PTM.

Average number of Extracted Terms used RFD			Average $weight(t)$ before revision			Average $weight(t)$ after revision			Terms extracted from D^+ used PTM	
$\#T^+$	$\#G$	$\#T^-$	$w(T^+)$	$w(G)$	$w(T^-)$	$w(T^+)$	$w(G)$	$w(T^-)$	$\#T$	$w(T)$
23.54	22.36	231.78	2.84211	1.40038	0.32031	4.15839	1.40038	-0.55127	156.9	1.45210

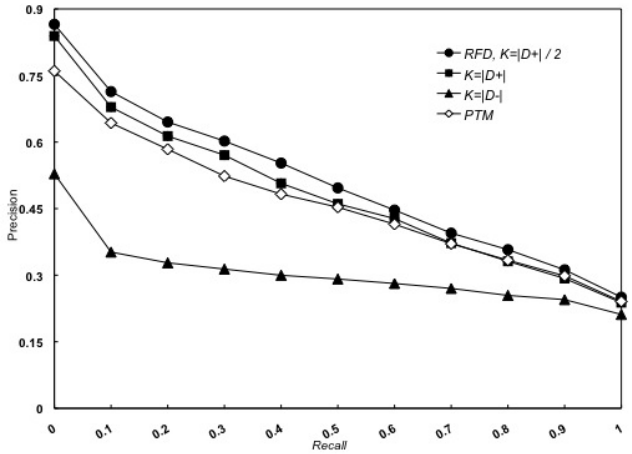

Figure 2: Comparison results of used different value for K in RCV1 dataset.

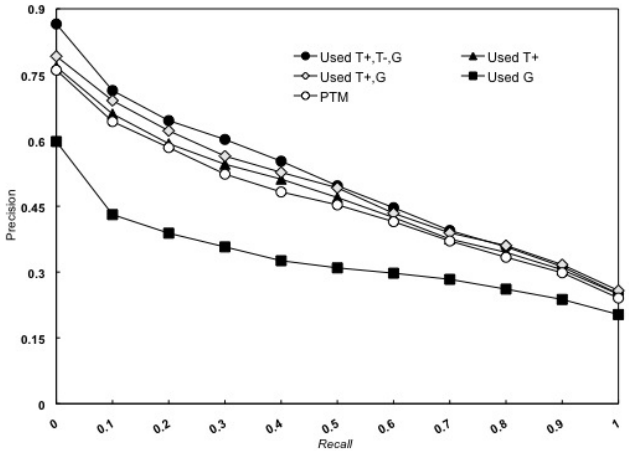

Figure 3: Comparison results of used different group of terms in all assessing topics.

sues of offender selection, the classification rules and noises, and recommendations.

6.4.1 Offender Selection

The positive feedback is more important than the negative one since the objective of model is for relevance feature discovery. However, we believe that negative feedback contains some information that can help to improve effectiveness of relevance feature discovery. The obvious problem for using negative documents is that most of the negative documents are noisy for a given topics because the very large resources of negative information. Therefore, it is necessary to choose some useful negative documents (or called offenders here) firstly to balance the amounts of terms in the three categories.

Technically, selecting large number of negative documents as offenders will weaken the importance of the features of T^+ (or G). Table 5 shows the average numbers of positive documents, negative documents and offenders in the training sets for the 50 assessing topics by using different values of K. From this table, we can see that the larger value of K is, the less percentages of features or weights in T^+ and G are (please note the significant increase of terms or weights in T^-).

Table 5 and Figure 2 also show the performances for different settings of K for offender selection for all 50 assessing topics. If we use all negative documents, the performance is even worse than PTM largely. Another advantage of offender selection is to reduce the space of negative relevance feedback. Table 5 clearly illustrates that only$15.8\% = \frac{6.54}{41.3}$ of the negative documents are selected as offenders.

In summary, the experimental results support the strategy of offender selection used for the proposed model. We can conclude that the proposed method for offender selection in RFD meets the design objectives.

6.4.2 Classification Rules

Low-level terms are grouped based on the specificity function and the classification rules. Specificity function describes how is the term related to the interested topic, and two thresholds have been used to decide the category of terms. Table 6 shows some possible values for the two thresholds θ_2 and θ_1. For the revision

of the weights of the low-level terms, the proposed model first classifies terms into three categories, given a distinct advantage.

Table 7 shows the average number of terms extracted by the proposed model and PTM. For each topic, the average number of terms that PTM extracted from positive documents is 156.9, and all those terms are used as one group. However, there are many noisy terms in the group to impose restrictions on the effectiveness. After using the proposed model to group the terms into three categories, the number of terms in both the positive specific category and the general category is reduced to $45.9 = 23.54 + 22.36$, that is, only 29.25% retained in RFD. Table 7 shows clearly that about $70.75\% = 100\% - 29.25\%$ of extracted terms from positive documents are possible noisy terms.

From the statistical information in Table 7, the percentage of general terms is $48.71\% = \frac{22.36}{22.36+23.54}$ comparing with the terms in the positive specific category. General terms frequently appear not only in positive documents, but also in some negative documents, because these negative documents may describe to some extent of the topic that users want. To reduce more the side effects of using general terms in relevance feature discovery, the proposed method adds negative specific terms into the low-level features. After revision, from Table 7, in the average, $120.78 = (23.54 + 22.36 + 231.78) - 156.9$ new negative features are added into T^-, and they are assigned weight -0.55127 (see Table 7) on average after revision.

6.4.3 *Specificity vs Exhaustivity*

Normally, we believe that positive specific terms (with large *specificity*) are more interesting than general terms (with large *exhaustivity*) for a given topic. As shown in Table 7, before revision, $66.99\% = \frac{2.84211}{1.40038+2.84211}$ weights are distributed to the specific positive terms, and 33.01% weights are distributed to general terms, that is, the classifications rules proposed in this paper looks reasonable. After revision, the proposed model also increases the weights of positive specific terms. The weight distributions change to $25.19\% = \frac{1.40038}{1.40038+4.15839}$ for the general term and 74.81% for the positive specific terms.

Increasing the weights of the low-level terms in positive specific category will reduce the side effect of the general terms. As shown in Figure 3, the use of positive specific terms only (Used T^+) can give much better result than the use of the only general terms (Used G). It is also recommended to use both positive specific terms and the general terms that can significantly improve the effectiveness (Used T^+, G). Although the general terms may frequently appear in negative documents, they are still important for the relevance feature discovery because they describe some of the extend to the given topic.

Moreover, there are many terms, for example $111 = 156.9 - (23.54 + 22.36)$ in Table 7, are moved to the negative specific category. In order to reduce the side effect of those terms, the proposed model reduce the weight of the low-level terms in the negative specific category T^-. The terms in the negative specific category are assigned weight -0.55127 on average after revision. In this way, these negative specific terms could reduce the side effects of general terms more if both general terms and negative specific terms appear in negative documents, because now only $16.96\% = \frac{1.40038-0.55127}{4.15839+1.40038-0.55127}$ weights could be distributed to the general terms, considering that positive specific terms get weight 4.15839 on average and general terms get $1.40038 - 0.55127$ on average.

In summary, the use of negative relevance feedback is very significant for the relevance feature discovery. It can balance the percentages of the specific terms and the general terms in order to reduce noises. The experimental results show that roughly we can choose the same amount of the positive specific terms and the general term, and assign large weights to the positive specific terms.

7. CONCLUSIONS

This paper presents an innovative approach for relevance feature discovery. It introduces a method to select negative documents (or called offenders) that are closed to the extracted features in the positive documents. It also proposes an approach to revise low-level features (terms) based on both their appearances in the higher-level features (patterns) and their categories (the positive specific category, general category and negative specific category). Compared with the state-of-the-art models, the experiments on RCV1 and TREC topics demonstrate that the effectiveness of relevance feature discovery can be significantly improved by the proposed approach.

Negative patterns could be useful for revising positive features in training set. However, whether negative patterns can largely improve accuracy is still an open question. This paper recommends to classify low-level terms into three categories in order to largely improve the performance of the revision. The proposed approach would be promising for mining relevance features in text documents.

Acknowledgements

This paper was partially supported by an ARC Discovery Grant from Australian Research Council (Project ID: DP0988007). We would like to thank the reviewers, especially Prof. Bruce Croft (our shepherd) for their constructive comments and suggestions.

8. REFERENCES

[1] H. Ahonen, O. Heinonen, M. Klemettinen, and A. I. Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In *Proc. of the Advances in Digital Libraries Conference*, USA, pages 2–11, 1998.

[2] C. Buckley. Evaluating evaluation measure stability. *Machine Learning*, 40:33–40, 2000.

[3] C. Buckley and G. Salton and J. Allan. The Effect of Adding Relevance Information in a Relevance Feedback Environment. In *Proc. of SIGIR'94*, pages 292-300, 1994.

[4] G. Cao and J. Nie and J. Gao and S. Robertson. Selecting good expansion terms for pseudo-relevance feeback. In *Proc. of SIGIR'08*, pages 243–250, 2008.

[5] Y. Cao and J. Xu and T.-Y. Liu and H. Li and Y. Huang and H.-W. Hon. Adapting Ranking SVM to Document Retrieval. In *Proc. of SIGIR'06*, pages 186–193, 2006.

[6] Bruce Croft and D. Metzler and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley, 2009.

[7] A. Dasgupta, P. Drineas, and B. Harb. Feature selection methods for text classification. In *Proc. of KDD'07*, pages 230–239, 2007.

[8] X. Geng and T.-Y. Liu and T. Qin. Query Dependent Ranking Using K-Nearest Neighbor. *In Proc. of SIGIR'08*, pages 115–122, 2008.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[10] Y. Huang and S. Lin. Mining sequential patterns using graph search techniques. In *Proc. of 27th Annual International Computer Software and Applications Conference*, pages 4–9, 2003.

[11] G. Ifrim, G. Bakir, and G. Weikum. Fast logistic regression for text categorization with variable-length n-grams. In *Proc. of KDD'08*, pages 354–362, 2008.

[12] N. Jindal and B. Liu. Identifying comparative sentences in text documents. In *Proc. of SIGIR'06*, pages 244–251, 2006.

[13] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proc. of ICML'97*, pages 143–151, 1997.

[14] S. Jones and S. WalkervvKaren and S. E. Robertson. A probabilisticmodel of information retrieval: Development and comparative experiments. *Information Processing & Management* 36(6): 779-808, 2000.

[15] V. Lavrenko and W.B. Croft. Relevance-based language models. In *Proc. of SIGIR'01*, pages 120–127, 2001.

[16] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proc. of SIGIR'92*, pages 37–50, 1992.

[17] D. D. Lewis and Y. Yang and T. G. Rose and F. Li. RCV1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361-397, 2004.

[18] X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. In *Proc. of IJCAI'03*, pages 587–594, 2003.

[19] Y. Li and S.-T. Wu and Y. Xu. Deploying association rules on hypothesis spaces. In *Proc. of International Conference on Computioonal Intelligence for Modelling, Control and Automation*, Australia, pages 769-778, 2004.

[20] 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.

[21] Y. Li, X. Zhou, P. Bruza, Y. Xu, and R. Y. Lau. A two-stage text mining model for information filtering. In *Proc. of CIKM'08*, pages 1023–1032, 2008.

[22] X. Ling, Q. Mei, C. Zhai, and B. Schatz. Mining multi-faceted overviews of arbitrary topics. In *Proc. of KDD'08*, pages 497–505, 2008.

[23] C. D. Manning and P. Raghavan and H. Schutze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[24] D. Metzler and W.B. Croft. Latent concept expansion using Markov random fields. In *Proc. of SIGIR'07*, pages 311–318, Amsterdam, Netherlands, 2007.

[25] R. K. Pon, A. F. Crdenas, D. Buttler, and T. Critchlow. Tracking multiple topics for finding interesting articles. In *Proc. of KDD'07*, pages 560–569, San Jose, California, USA, 2007.

[26] J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *Proc. of SIGIR'98*, pages 275–281, 1998.

[27] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, and H. Li. Ranking with multiple hyperplanes. In *Proc. of SIGIR'07*, pages 279–286, 2007.

[28] S. Robertson and I. Soboroff. The TREC 2002 filtering track report. In *TREC'02*, 2002.

[29] S. E. Robertson, H. Zaragoza, and M. J. Taylor. Simple bm25 extension to multiple weighted fields. In *Proc. of CIKM'04*, pages 42–49, 2004.

[30] J. J. Rocchio. Relevance feedback in information retrieval. In Salton, Gerard (ed., 1971), *The SMART Retrieval System Experiments in Automatic Document Processing*, Prentice Hall, pages 313–323, 1971.

[31] S. Scott and S. Matwin. Feature engineering for text classification. In *Proc. of ICML'99*, pages 379–388, 1999.

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

[33] M. Seno and G. Karypis. Slpminer: An algorithm for finding frequent sequential patterns using length-decreasing support constraint. In *Proc. of ICDM'02*, pages 418–425, 2002.

[34] S. Shehata, F. Karray, and M. Kamel. A concept-based model for enhancing text categorization. In *Proc. of KDD'07*, pages 629–637, 2007.

[35] I. Soboroff and S. E. Robertson. Building a filtering test collection for trec 2002. In *Proc. of SIGIR'03*, pages 243–250, 2003.

[36] F. Song and W.B. Croft. A general language model for information retrieval. In *Proc. of CIKM'99*, pages 316–321, 1999.

[37] X. Tao and Y. Li and N. Zhong. A Personalized Ontology Model for Web Information Gathering. Accepted by *IEEE Transactions on Knowledge and Data Engineering*, 1 Dec 2009.

[38] C. J. van RIJSBERGEN. *Information Retrieval*. London, Butterworths, 1979.

[39] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *Proc. of SIGIR'08*, pages 219–226, 2008.

[40] S.-T. Wu, Y. Li, and Y. Xu. Deploying approaches for pattern refinement in text mining. In *Proc. of ICDM'06*, pages 1157–1161, 2006.

[41] S.-T. Wu, Y. Li, Y. Xu, B. Pham, and P. Chen. Automatic pattern-taxonomy extraction for web mining. In *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, pages 242–248, 2004.

[42] Y. Xu and Y. Li. Generating concise association rules. In *Proc. of CIKM'07*, pages 781–790, 2007.

[43] Z. Xu and R. Akella. Active Relevance Feedback for Difficult Queries In *Proc. of CIKM'08*, pages 459–468, 2008.

[44] J. Xu and W. B. Croft. Improving the Effectiveness of Information Retrieval with Local Context Analysisn. *ACM Transactions on Information Systemsl*, 18(1):79–112, 2000.

[45] G. Xue and D. Xing and Q. Yang and Y. Yu. Deep Classification in Large-scale Text Hierarchies. In *Proc. of SIGIR'08*, pages 619–626, 2008.

[46] X. Yan, J. Han, and R. Afshar. Clospan: mining closed sequential patterns in large datasets. In *Proc.of SIAM International Conference on Data Mining* , pages 166–177, 2003.

[47] X. Yan and H. Cheng and J. Han and D. Xin. Summarizing itemset patterns: a profile-based approach. In *Proc. of KDD*, pages 314–323, 2005.

[48] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90, 1999.

[49] M. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 40:31–60, 2001.

[50] C. Zhai and J. Lafferty. Model-based feedback in language modeling approach to information retrieval. In *Proc. of CIKM'01*, pages 403–410, 2001.

[51] S. Zhu and X. Ji and W. Xu and Y. Gong. Multi-labelled Classification Using Maximum Entropy Method. In *Proc. of SIGIR'05*, pages 274–281, 2005.