Inf Retrieval (2011) 14:237–256
DOI 10.1007/s10791-010-9154-4

WEB MINING FOR SEARCH

A pattern mining approach for information filtering systems

Yuefeng Li · Abdulmohsen Algarni · Yue Xu

Received: 10 May 2010 / Accepted: 8 November 2010 / Published online: 14 December 2010

Abstract It is a big challenge to clearly identify the boundary between positive and negative streams for information filtering systems. Several attempts have used negative feedback to solve this challenge; however, there are two issues for using negative relevance feedback to improve the effectiveness of information filtering. The first one is how to select constructive negative samples in order to reduce the space of negative documents. The second issue is how to decide noisy extracted features that should be updated based on the selected negative samples. This paper proposes a pattern mining based approach to select some offenders from the negative documents, where an offender can be used to reduce the side effects of noisy features. It also classifies extracted features (i.e., terms) into three categories: positive specific terms, general terms, and negative specific terms. In this way, multiple revising strategies can be used to update extracted features. An iterative learning algorithm is also proposed to implement this approach on the RCV1 data collection, and substantial experiments show that the proposed approach achieves encouraging performance and the performance is also consistent for adaptive filtering as well.

Keywords Pattern mining · Relevance feedback · Information filtering

1 Introduction

Traditional information filtering (IF) models were developed based on a term-based user profile approach (see [15, 20, 23]). The advantage of term-based profiles is efficient computational performance as well as mature theories for term weighting, which have

Y. Li (✉) · A. Algarni · Y. Xu
Discipline of Computer Science, Queensland University of Technology, Brisbane, QLD 4001, Australia
e-mail: y2.li@qut.edu.au

A. Algarni
e-mail: a1.algarni@qut.edu.au

Y. Xu
e-mail: yue.xu@qut.edu.au

emerged over the last couple of decades from the information retrieval (IR) and machine learning communities. However, term-based profiles suffer from the problems of polysemy and synonymy. As IF systems are sensitive to data sets, it is still a challenging issue to significantly improve the effectiveness of IF systems.

Over the years, people have often held the hypothesis that phrases would perform better than words, as phrases are more discriminative and arguably carry more "semantics". This hypothesis has not fared too well in the history of IR [11, 27, 28] in the beginning. Recently, language modeling approaches went beyond the term based model that underlies BM25 by considering term dependencies in phrases (N-grams) for information retrieval [18, 35]. Although phrases are less ambiguous and more discriminative than individual terms, the likely reasons for the discouraging performance include: (1) phrases have inferior statistical properties to words since they have low frequency of occurrence, (2) the theory of computing probabilities based on term dependencies is not practical, (3) some language model-based feedback methods cannot naturally handle negative feedback, and (4) there are large numbers of redundant and noisy phrases among them.

To overcome the limitations of term-based approaches, pattern mining based techniques have been used for information filtering since data mining has developed some techniques (e.g., maximal patterns, closed patterns and master patterns) for removing redundant and noisy patterns. One special filtering task was to extract usage patterns from Web logs [4, 47]. Other promising techniques were pattern taxonomy models (PTM) [32, 37] that discovered closed sequential patterns in text documents, where a pattern was a set of terms that frequently appeared in paragraphs.

Pattern based approaches have shown encouraging improvements on effectiveness [36]. However, two challenging issues have arisen when pattern mining techniques were introduced for IF systems. The first one is how to deal with low frequency patterns because the measures used for data mining (e.g., "support" and "confidence") to learn the patterns turn out be not suitable in the filtering stage [15]. The second issue is how to effectively use negative feedback to revise extracted features (including patterns and terms) for information filtering.

Many people believe that there are plenty negative information available and negative documents are very useful because they can help users to search for accurate information [35]. However, whether negative feedback can indeed largely improve filtering accuracy is still an open question. The existing methods of using both positive and negative feedback for IF can be grouped into two approaches. The first approach is to revise terms that appear in both positive samples and negative samples (e.g., Rocchio based models and SVM [23] based filtering models). This heuristics 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 samples and negative samples (e.g., probabilistic models [2], and BM25 [23]). However, usually people view terms in multiple perspectives when they attempt to find what they want. They normally use two dimensions ("specificity" and "exhaustivity") for deciding the relevance of documents, paragraphes or terms. For example, "JDK" is a specific term for "Java Language", and "LIB" is more general than "JDK" because it is also frequently used for C and C++ as well.

Based on this observation, this paper proposes a pattern mining based approach for using both positive and negative feedback. It firstly extracts an initial list of terms from positive documents and selects some constructive negative documents (or called offenders). It then extracts terms from negative patterns in selected negative documents. It also classifies all terms into three categories: the positive specific terms, general terms, and negative specific terms. In this way, multiple revising strategies are used for terms in

different categories. In the implementation, it recommends to increment positive specific terms' weights only and declines negative specific terms' weights based on their occurrences in discovered negative patterns. Substantial experiments show that the proposed approach achieves exciting performance.

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 pattern taxonomy mining. Section 4 introduces the equations for evaluating term weights based on discovered patterns. Section 5 describes the proposed method of using negative feedback. The empirical results and discussion are reported in Sect. 6, and the last section describes concluding remarks.

2 Related work

Different from IR systems, IF systems were commonly personalized to support long-term information needs of users [3]. The main distinct difference between IR and IF was that IR systems used "queries" but IF systems used "user profiles". The tasks of the filtering included adaptive filtering, and batch or routing filtering. In this paper, the focus is on the breakthrough for batch or routing filtering. Adaptive filtering involves feedback to dynamically adapt IF systems [9, 17, 33, 42, 44]. The popular way is to update training sets in a batch classifier fashion. In this paper, we also evaluate the performance of the proposed approach for adaptive filtering.

Normally, IF systems tended to learn a map $rank : \mathbb{D} \rightarrow \mathbb{R}$ such that $rank(d)$ corresponded to the relevance of a document d, where $\mathbb{D}$ denoted a set of documents, $\mathbb{R}$ was the set of real numbers. In [20], *rank* was divided into two functions, such that $rank = f_1 \circ f_2$, where f_1 ($f_1 : D \rightarrow \{C_1, \ldots, C_m\}$) and f_2 ($f_2 : \{C_1, \ldots, C_m\} \rightarrow \mathbb{R}$) were maps, respectively; and $C_1, C_2, \ldots, C_m$ were clusters. This method used a set of clusters based on a kind of classification method, e.g., the neural network [19]. The aim of the filtering track in TREC [23] was to measure the ability of IF systems to build profiles using sets of training documents to separate relevant and non-relevant documents. The basic term-based IF models used in TREC 2002 were SVM, Rocchio's algorithm, probabilistic models, and BM25.

Feedback techniques are frequently used in IR community to improve the accuracy of filtering. Normally, there are different strategies for considering users feedback information for information retrieval. They are relevance feedback, pesudo-relevance feedback, implicit feedback and negative feedback [6, 29, 34, 38]. One of the common objectives of these strategies is to design IR models in order to obtain more accurate term weights based on user feedback for a given query.

Term-based models are most widely used approaches. A term-based model is based on the bag of words or N-grams, which uses terms as elements and evaluates term weights based on terms' appearances or frequencies in feedback. For example, Rocchio-style classifiers [12], ranking SVM [22]; and BM25 for structured documents [25] are popular IF systems. They can also naturally handle both positive and negative feedback information. However, the research on term-based models has arguably hit somewhat of a wall in terms of effectiveness improvement possibly due to the ambiguity problem mentioned earlier. In addition, modeling the real dependencies between terms is very difficult.

Language models have been developed for considering term dependencies. In a language model, the key elements are the probabilities of word sequences which include both terms and phrases (or sentences) [31]. They are often approximated by N-gram models,

such as Unigram, Bigram or Trigram, for considering term dependencies easily. Language modeling approaches include model-based methods, and relevance models [18]. The former finds models that can best describe the features in positive documents while considering a background model [45]. The later tries to model the notation of relevance in a more generalized level [10]. Language modeling approaches have been well developed for information retrieval, especially for query expansion techniques [18, 39, 35]. They are also quite effective for exploiting positive feedback information. However, they cannot naturally handle negative feedback.

Pattern mining has been extensively studied in data mining communities for many years. A variety of efficient algorithms such as Apriori-like algorithms [1], PrefixSpan [21], and FP-tree [5] have been proposed. These research works have mainly focused on developing efficient mining algorithms for discovering patterns in databases. Usually, the existing data mining techniques return numerous discovered patterns (e.g., sets of terms) from a training set, but large numbers of them are redundant patterns [40]. Nevertheless, the challenging issue is how to effectively deal with the large amount of discovered patterns and terms with a lot of noises.

Closed patterns have turned out to be a promising alternative to phrases [7, 32] because patterns enjoy good statistical properties like terms. To effectively use closed patterns for information filtering, closed sequential patterns have been used in pattern taxonomy models (PTM) [32, 36, 37], which deployed closed sequential patterns 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 probabilistic models, Rocchio based method and N-gram. The similar research also appeared in [41] 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.

These approaches introduced data mining techniques to information filtering; however, too many noisy patterns adversely affect PTM systems [15]. The major research issue is how to use both positive and negative feedback to significantly reduce the effects of noisy patterns. Traditional data mining techniques can only achieve a little progress for the effectiveness because they can only discuss this problem at the pattern level. This paper starts to consider human being's perspective about relevance and uses a two-dimension concept to classify terms into three groups: positive specific terms, general terms and negative specific terms. In this perspective, term weights can be evaluated accurately based on their appearances in both positive patterns and negative patterns.

Our conference paper [13] is the first study on the problem of mining negative relevance feedback for information filtering. In this paper, we extend previous study by adding more examples, discussing more related research works, and extending the experiments for discussing the proposed iterative learning algorithm and statistic analysis. We also conducted some new experiments for using the proposed approach on adaptive filtering.

3 Pattern taxonomy mining

In this paper, we assume that all documents are split in paragraphs. So a given document d yields a set of paragraphs $PS(d)$. Let D be a training set of documents, which consists of a set of positive documents, D^+; and a set of negative documents, D^-.

Let $T = \{t_1, t_2, \ldots, t_m\}$ be a set of terms (or keywords) which are extracted from the set of positive documents, D^+.

3.1 Frequent and closed patterns

Given a *termset* X, a set of terms, in document d, $\ulcorner X \urcorner$ is used to denote the covering set of X for d, which includes all paragraphs $dp \in PS(d)$ such that $X \subseteq dp$, i.e., $\ulcorner X \urcorner = \{dp | dp \in PS(d), X \subseteq dp\}$. Its *absolute support* is the number of occurrences of X in $PS(d)$, that is $sup_a(X) = |\ulcorner X \urcorner|$. Its *relative support* is the fraction of the paragraphs that contain the pattern, that is, $sup_r(X) = \frac{|\ulcorner X \urcorner|}{|PS(d)|}$. A termset X is called *frequent pattern* if its sup_a (or sup_r) $\geq min\_sup$, a minimum support.

Table 1 lists a set of paragraphs for a given document d, where $PS(d) = \{dp_1, \ldots, dp_6\}$, and duplicate terms are removed. Let $min\_sup = 3$ giving rise to ten frequent patterns which are illustrated in Table 2. Normally not all frequent patterns are useful [32, 40]. For example, pattern $\{t_3, t_4\}$ always occurs with term t_6 in paragraphs (see Table 1); therefore, we want to keep the larger pattern only.

Given a termset X, its covering set $\ulcorner X \urcorner$ is a subset of paragraphs. Similarly, given a set of paragraphs $Y \subseteq PS(d)$, we can define its *termset*, which satisfies

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

The closure of X is defined as follows:

$$Cls(X) = termset(\ulcorner X \urcorner).$$

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

Table 1 A set of paragraphs

Parapgraph	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	$t_1\, t_2\, t_6\, t_7$

Table 2 Frequent patterns and covering sets

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

Let X be a closed pattern. We have

$$sup_a(X_1) < sup_a(X) \tag{1}$$

for all pattern $X_1 \supset X$.

3.2 Pattern taxonomy

Patterns can be structured into a taxonomy by using the *is-a* (or *subset*) relation and closed patterns. For example, Table 2 contains ten frequent patterns; however, it includes only three closed patterns: $\langle t_3, t_4, t_6\rangle$, $\langle t_1, t_2\rangle$, and $\langle t_6\rangle$. Simply, a pattern taxonomy is described as a set of pattern-absolute support pairs, for example $\mathrm{PT} = \{\langle t_3, t_4, t_6\rangle_3, \langle t_1, t_2\rangle_3, \langle t_6\rangle_5\}$, where non-closed patterns are pruned. After pruning, some direct "is-a" retaliations may be changed, for example, pattern $\{t_6\}$ would become a direct sub-pattern of $\{t_3, t_4, t_6\}$ after pruning non-closed patterns $\langle t_3, t_6\rangle$ and $\langle t_4, t_6\rangle$.

Smaller patterns in the taxonomy, for example pattern $\{t_6\}$, are usually more general because they could be used frequently in both positive and negative documents; and larger patterns, for example pattern $\{t_3, t_4, t_6\}$, in the taxonomy are usually more specific since they may only used in positive documents.

3.3 Closed sequential patterns

A sequential pattern $s = \langle t_1, \ldots, t_r\rangle$ ($t_i \in T$) is an ordered list of terms. A sequence $s_1 = \langle x_1, \ldots, x_i\rangle$ is a sub-sequence of another sequence $s_2 = \langle y_1, \ldots, y_j\rangle$, 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 simply say patterns for sequential patterns.

Given a pattern (an ordered *termset*) X in document d, $\ulcorner X \urcorner$ 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., $\ulcorner X \urcorner = \{ps | ps \in PS(d), X \sqsubseteq ps\}$. Its *absolute support* and *relative support* are defined as the same as for the normal patterns.

A sequential pattern X is called *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 Deploying patterns on terms

The evaluation of term supports (weights) in this paper is different from the term-based approaches. For a term based approach, the evaluation of a given term's weight is based on its appearance in documents. For pattern mining, terms are weighted according to their appearance in discovered patterns.

To improve the efficiency of the pattern taxonomy mining, *SPMining*(D^+, *min_sup*) algorithm [32], was proposed (also used in [15, 37]) to find closed sequential patterns for all document $d \in D^+$, which used the well-known Apriori property in order to reduce the searching space. For all positive document $d \in D^+$, the *SPMining* algorithm discovered all closed sequential patterns based on a given *min_sup*.

Let SP_1, SP_2,..., $SP_{|D^+|}$ be the sets of discovered closed sequential patterns for all document $d_i \in D^+ (i = 1, \cdots, |D^+|)$. For a given term t, its support in these discovered patterns can be described as follows:

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

Table 3 illustrates a real example of pattern taxonomy for a set of positive documents $D^+ = \{d_1, d_2, \cdots, d_5\}$. For example, term *global* appears in three documents (d_2, d_3 and d_5). Therefore, its support can be calculated based on patterns in the three documents's pattern taxonomies:

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

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

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

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

5 Mining negative feedback

In general, the concept of relevance is subjective; and normally people can describe the relevance of a topic (or document) in two dimensions: the specificity and 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 the two dimensions for IF systems. In this section, we first discuss how to use the two dimensions for understanding the different roles of the selected terms. We also presents an algorithm for both negative document selection and term weight revision.

Table 3 Example of sets of discovered closed sequential patterns in pattern taxonomies, where the minimum absolute support is 2

Doc.	Pattern taxonomies	Sets of discovered closed sequential patterns (SP)
d_1	$PT_{(1,1)}$	$\{\langle carbon \rangle, \langle carbon, emiss \rangle\}$
	$PT_{(1,2)}$	$\{\langle air, pollut \rangle\}$
d_2	$PT_{(2,1)}$	$\{\langle greenhous, global \rangle\}$
	$PT_{(2,2)}$	$\{\langle emiss, global \rangle\}$
d_3	$PT_{(3,1)}$	$\{\langle greenhous \rangle\}$
	$PT_{(3,2)}$	$\{\langle global, emiss \rangle\}$
d_4	$PT_{(4,1)}$	$\{\langle carbon \rangle\}$
	$PT_{(4,2)}$	$\{\langle air \rangle, \langle air, antarct \rangle\}$
d_5	$PT_{(5,1)}$	$\{\langle emiss, global, pollut \rangle\}$

5.1 Specific and general terms

Formally, let DP^+ be the union of all discovered positive patterns of pattern taxonomies of D^+, and DP^- be the union of all discovered negative patterns of pattern taxonomies of D^-, where a closed sequential pattern of D^- is called negative pattern. Given a term $t \in T$, its *exhaustivity* is the number of discovered patterns in both DP^+ and DP^- that contain t, and its *specificity* is the number of discovered patterns in DP^+ but not in DP^- that contain t. Based on this understanding, in this paper we classify terms into three groups. We call a term a *general term* if it appears in both positive patterns and negative patterns. We also call terms positive (or negative) *specific terms* if they appear only in patterns discovered in positive (or negative) documents only.

Based on the above discussion, we have the following definitions for the set of general terms GT, the set of positive specific terms T^+, and the set of negative specific terms T^-:

$$GT = \{t|(\exists p_1 \in DP^+) \wedge (\exists (p_2 \in DP^-) \Rightarrow t \in (p_1 \cap p_2)\},$$
$$T^+ = \{t|t \notin GT, \exists (p \in DP^+) \Rightarrow t \in p\}, \text{ and}$$
$$T^- = \{t|t \notin GT, \exists (p \in DP^-) \Rightarrow t \in p\}.$$

It is easy to verify that $GT \cap T^+ \cap T^- = \emptyset$. Therefore, (GT, T^+, T^-) is a partition of all terms in patterns.

To describe user profiles for a given topic, normally we believe that specific terms are very useful for the topic in order to distinguish to other topics. However, some experimental results show that using only specific terms are not good enough to improve the performance of information filtering because user information needs cannot simply be covered by documents that only contain the specific terms. Therefore, the best way is to use the specific terms mixed with some of the general terms.

5.2 Strategies of revision

After we can classify terms into three categories, we firstly show the basic process of revising discovered features in the training set. This process can help readers to understand the proposed strategies for revising discovered features in different categories.

The process first extracts initial features in the positive documents in the training set, which include terms and patterns. It then selects some negative samples (or called offenders) in the set of negative documents in the training set. It also extracts negative features, including both terms and negative patterns, from the selected negative documents using the same pattern mining technique as used for the feature extraction in positive documents. In addition, it revises the initial features and obtains revised features. The process can be repeated for several times as follows: selecting negative documents, extracting negative features and revising revised features.

Algorithm *NFMining*(D) describes the details of the strategies of the revision, where we assume that the number of negative documents is greater than the number of positive documents. For a given training set $D = \{D^+, D^-\}$, we assume that the initial features, $\langle T, DP^+, DP^- \rangle$, have been extracted from positive documents D^+ before we start the algorithm, where we let $DP^- = \varnothing$. We also let the experimental parameter $\alpha = -1$ that will be used for calculating weights of terms in negative patterns.

Step 1 initializes the set of general terms GT, the set of positive specific terms T^+ and the set of negative specific terms T^-, where *loop* is used to control the times of the revision. Step 2 and 3 calculate terms' weights for all term in T.

Step 4 and 5 rank documents in the set of negative documents, where if t is a negative specific term, its weight is the revised weight that calculates in step 10 and 11. The weight function can be described as follows:

$$weight(t) = \begin{cases} \text{its revising weight}, & \text{if } t \in T^- \\ support(t, D^+), & \text{otherwise} \end{cases}$$

Step 6 and 7 sort the negative documents based on documents' rank values, and select offenders, some negative documents. If a document's rank less than or equals to 0 that means this document is clearly negative to the system. A document has hight rank that means the document is an offender because it forces the system make mistake. The offenders are normally defined as the top-K negative documents in sorted D^- [14]. In this paper, we let $K = \lceil \frac{|D^+|}{3} \rceil$. In the first revision ($loop = 0$), we ignore the top-j negative documents for offender selection since the initial features only coming from positive documents and we believe that positive features are more important than negative features in the beginning, where $j = \lfloor \frac{|D^-|}{|D^+|} \rfloor$, the largest integer that less than or equals to $\frac{|D^-|}{|D^+|}$.

Step 8 and 9 extract negative features (DP^-, T_0) from selected negative documents D_3^-, where it calls algorithm *SPMining* (D_3^-, min_sup) to discover negative patterns DP^- and T_0 that includes all terms in patterns in DP^-.

Step 10 to 12 revise negative specific terms' weights. These steps will go through a loop for three times and the iteration is controlled by step 13. In each loop, when a specific negative term is extracted in the first time, the algorithm simply negatives its support obtained from the selected negative documents; otherwise, the algorithm cumulates its weight as follows:

$$weight(t) = \alpha \times support(t, D_3^-) + weight(t).$$

After three loops, the algorithm participates T into general terms GT and positive specific terms T^+ in step 14 and 15. It also revises positive specific terms' weights using the following equation in step 16 and 17:

$$weight(t) = weight(t) * (1 + \frac{|\{d|d \in D^+, t \in d\}|}{|D^+|})$$

At last, it updates T to include negative specific terms in step 18.

NFMining calls three times *SPMining* and the total negative documents used in the three times is $O(|D^+|)$; therefore, it takes the same computation time for mining patterns in selected negative documents as the *SPMining* does for mining patterns in positive documents. *NFMining* also takes times for sorting D^-, assigning weights to terms and partitioning terms into groups. The time complexity for these operations is $O(|D^-|(log^{(|D^-|)} + |T|) + |T|^2)$.

This algorithm consists of three loops for mining negative specific terms and the corresponding weights. For each loop, after finishing the loop, it is obvious that the number of negative specific terms, $|T^-|$, is not less than the number of negative specific terms before the loop, because of the operation, $T^- = T^- \cup (T_0 - T)$, in Step 12. We expect the three loops can produce enough negative specific terms in order to reduce the side effects of general terms. We will discuss more details for this question in Sect. 6.4.

6 Evaluation

In this section, we first discuss the data collection used for our experiments. We also describe the baseline models and their implementation. In addition, we present the experimental results and the discussion.

NFMining(D)

Input: A training set, $\{D^+, D^-\}$, parameter $\alpha = -1$;
extracted features $< T, DP^+, DP^- >$, $DP^- = \varnothing$;
support function and minimum support *min_sup*.

Output: Updated term set T and function *weight*.

Method:

1: $GT = \emptyset, T^+ = \emptyset, T^- = \emptyset$, $loop = 0$;
2: **foreach** $t \in T$ **do**
3: $weight(t) = support(t, D^+)$;
4: **foreach** $d \in D^-$**do**
5: $rank(d) = \Sigma_{t \in d \cap (T \cup T^-)} weight(t)$;
6: let $D^- = \{d_0, d_1, ..., d_{|D^-|-1}\}$ in descendent ranking order,
let $j = \lfloor \frac{|D^-|}{|D^+|} \rfloor$ if $loop = 0$, otherwise $j = 0$;
7: $D_3^- = \{d_i | d_i \in D^-, j \leq i < \lceil \frac{|D^+|}{3} \rceil + j\}$;
8: $DP^- = SPMining\,(D_3^-, min\_sup)$; //find negative patterns
9: $T_0 = \{t \in p \mid p \in DP^-\}$; // all terms in negative patterns
10: **foreach** $t \in (T_0 - T)$ **do**
11: **if** $(loop = 0)$ **then** $weight(t) = \alpha \times support(t, D_3^-)$
else $weight(t) = \alpha \times support(t, D_3^-) + weight(t)$;
12: $T^- = T^- \cup (T_0 - T)$, $loop++$;
13: **if** $loop < 3$ **then** goto step 4;
14: **foreach** $t \in T$ **do** //term partition
15: **if** $(t \in T^-)$ **then** $GT = GT \cup \{t\}$
else $T^+ = T^+ \cup \{t\}$;
16: **foreach** $t \in T^+$ **do**
17: $weight(t) = weight(t) * (1 + \frac{|\{d | d \in D^+, t \in d\}|}{|D^+|})$;
18: $T = T \cup T^-$;

6.1 Data

Reuters Corpus Volume 1 (RCV1) was 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 with total 806,791 documents. The document collection is divided into training sets and testing sets.

TREC (2002) has developed and provided 100 topics for the filtering track aiming at building a robust filtering system. The topics are of two types: (1) A first set of 50 topics are developed by the assessors of the National Institute of Standards and Technology

(NIST) (i.e., assessor topics); The relevance judgements have been made by assessor of NIST. (2) A second set of 50 topics have been constructed artificially from intersections of pairs of Reuters categories (i.e., intersection topics) [30].

Difference from the assessor topics, the relevance judgements have been made by machine learning methods not by human being for intersection topics. The assessor topics are more reliable and the quality of the intersection topics is not quite good [23, 30]. For this reason, we use the all 50 assessor topics in this paper.

Documents in the RCV1 collection 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 term by applying the Porter Stemming algorithm are conducted [16].

6.2 Baseline models and setting

In this paper, we select three term-based baseline models because they are frequently used for both positive and negative documents. They are a Rocchio model, a BM25 based IF model, and a SVM based model. The PTM model is also used to measure the performance of using negative feedback for pattern mining. In this paper, the proposed approach is called Negative PaTtern Mining model (N-PTM), which firstly discovers sequential closed patterns from positive documents, deploys discovered patterns on their terms. Then, it discovers negative patterns from negative documents to group and revise the extracted features from positive documents as shown in Sect. 5.

The Rocchio algorithm [26] has been widely adopted in the areas of text categorization and information filtering. It can be used to build the profile for representing the concept of a topic which consists of a set of relevant (positive) and irrelevant (negative) documents. The Centroid $\vec{c}$ of a topic can be generated as follows:

$$\alpha \frac{1}{|D^+|} \sum_{\vec{d} \in D^+} \frac{\vec{d}}{||\vec{d}||} - \beta \frac{1}{|D^-|} \sum_{\vec{d} \in D^-} \frac{\vec{d}}{||\vec{d}||}$$

There are two sets of setting for α and β: $\alpha = 16$ and $\beta = 4$; and $\alpha = \beta = 1.0$. We tested both sets and found $\alpha = \beta = 1.0$ was the best set. So, we use $\alpha = \beta = 1.0$ in the above equation.

BM25 [8, 24] is the one of state-of-the-art retrieval functions used in document retrieval. The term weights are estimated using the following BM25 based equation:

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

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).

Information filtering can also be regarded as a special instance of text classification [28]. SVM is a statistical method that can be used to find a hyperplane that best separates two

classes. SVM achieved the best performance on the Reuters-21578 data collection for document classification [43]. The decision function in SVM is defined as:

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

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

$$\forall_i : \alpha_i \geqslant 0 \quad \text{and} \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2}$$

To compare with other baseline models, we tried to use SVM to rank documents rather than to make binary decisions. For this purpose, threshold b can be ignored. We also believe that the positive documents in the training set should have the same importance to user information needs because the training set was only simply divided into positive documents and negative documents. So 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. 2. 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

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

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

PTM model is also selected as one of the baselines models because we want to verify that mining negative feedback can significantly improve the performance of PTM. The maximum size of the term set T is 4000 for PTM. We also set $min\_sup = 0.2$ (relative support) for both PTM and N-PTM.

The performance of PTM was based on the number of closed patterns that were decided by a minimum support [36]. If the minimum support is very small, many noisy patterns can be introduced to the system; however, if it is very big then many useful patterns may be missed out. For RCV1, the total number of frequent sequential patterns is 36,202 that includes 28,733 closed patterns if $min\_sup = 0.2$. PTM can remove 20% of the frequent patterns if $min\_sup = 0.2$. In this paper, we use a fixed minimum support value, $min\_sup = 0.2$, suggested by [36].

6.3 Results

The effectiveness was measured by four different means: The F-beta (F_β) measure, Mean Average Precision (MAP), the break-even point (b/p), and Interpolated Average Precision (IAP) on *11-points*.

F_β is calculated by the following function:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

The parameter $\beta = 1$ is used in our study, which means that recall and precision is weighed equally. Mean Average precision is calculated by measuring precision at each relevant document first, and averaging precision over all topics. The *b/p* break-even point indicates the value at which precision equals recall. The larger a *b/p*, MAP, IAP or F_β-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).

Statistical method is also used to analyze the experimental results. The t-test assesses whether the means of two groups are statistically different from each other. The paired two-tailed t-test is used in this paper. If *DIF* represents the difference between observations, the hypotheses are: *Ho*: $DIF = 0$ (the difference between the two observations is 0). *Ha*: $DIF \neq 0$ (the difference is not 0). *N* is the sample size of group. The test statistic is *t* with $N - 1$ degrees of freedom (*df*). If the *p* value associated with *t* is low (<0.05), there is evidence to reject the null hypothesis. Thus, there is evidence that the difference in means across the paired observations is significant. The N-PTM model is compared with PTM, Rocchio, BM25, and SVM models for each variable *b/p*, *MAP*, *IAP*, $F_{\beta=1}$ over all the 50 topics, respectively.

6.3.1 N-PTM vs baseline models

Table 4 illustrates the results of all models against the five measures for all assessor topics. Compared with PTM which uses positive documents only, the proposed N-PTM model uses both positive and negative feedback. It is obvious that N-PTM is extremely better than PTM for all five measures. The proposed model N-PTM is also compared with term-based baseline models in Table 4 including Rocchio, BM25, and SVM, which also use both positive and negative feedback as well. The results of *11-points* on all assessor topics are reported in Fig. 1.

As shown in Table 4 and Fig. 1, the proposed new model (N-PTM) has achieved the best performance results for the assessor topics.

We also conducted the *t* test to compare the proposed model with all baseline models and the results are listed in Table 5. The percentage changes are shown in Table 6. Comparing with these baseline models, the proposed approach achieves excellent performance with 13.73% (max 17.34% and min 8.76%) average percentage change for all five measures.

These statistic results indicate that the proposed model is extremely statistically significant. Therefore, we conclude that mining negative relevance feedback for information filtering is an exciting achievement for pattern based approaches.

Table 4 Results for all assessor topics on RCV1

Model	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
N-PTM	0.5470	0.4718	0.4863	0.4631	0.5067
PTM	0.4960	0.4304	0.4436	0.4392	0.4641
BM25	0.4450	0.4074	0.4069	0.4140	0.4281
SVM	0.4530	0.4083	0.4092	0.4211	0.4353
Rocchio	0.4740	0.4201	0.4305	0.4299	0.4523

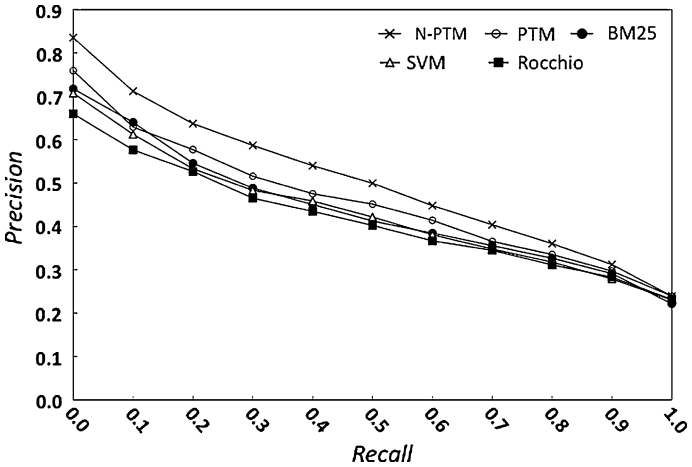

Fig. 1 Comparison between the proposed method and other approachs

Table 5 *P* value for all models comparing with N-PTM

Model	*Top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
PTM	0.027389945	0.002712978	0.00185474	0.002748289	0.000923675
BM25	0.000961513	0.002827644	0.000806368	0.000926226	0.000416951
SVM	0.000876284	0.001794258	0.000124716	0.00049229	0.000158923
Rocchio	0.009870197	0.012662994	0.008214849	0.010978002	0.007517726

Table 6 Percentage change over all baseline models

Model	*Top-20* (%)	*b/p* (%)	*MAP* (%)	$F\ (\%)_{\beta\ =\ 1}$	*IAP* (%)
PTM	10.28	9.62	9.64	5.45	9.18
BM25	22.92	15.81	19.53	11.86	18.36
SVM	20.75	15.57	18.85	9.99	16.40
Rocchio	15.40	12.32	12.97	7.75	12.03
AVG	17.34	13.33	15.23	8.76	13.99

In the training phase, it is obvious that N-PTM and PTM use more times than other term-based models because of mining patterns in paragraphes. However, for the time complexity in the testing phase, all models take $O(|T| \times |d|)$ for all incoming documents *d*. In our experiments, the number of terms used by all models for each topic is less than 300 in average. Therefore, there is no significant difference between these models on time complexity in the testing phase.

6.3.2 Adaptive filtering

In this section, we design some experiments for testing the adaptive performance of the proposed *N-PTM* model. We expect these experiments can achieve the consistent performance like the batch one in the last section.

Table 7 Adaptive N-PTM models for all assessor topics

Model	*Top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
N-PTM-1	0.570	0.479	0.496	0.473	0.517
N-PTM-2	0.567	0.486	0.512	0.483	0.529
N-PTM-3	0.575	0.481	0.504	0.475	0.524
N-PTM-4	0.562	0.497	0.514	0.481	0.536
N-PTM-5	0.541	0.462	0.487	0.464	0.507
N-PTM-6	0.543	0.466	0.490	0.467	0.513
Avg	0.560	0.478	0.501	0.474	0.521
chg% to Rocchio	12	7.66	10.1	6.28	9.45

Table 8 Adaptive Rocchio models for all assessor topics

Model	*Top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
Rocchio-1	0.525	0.444474	0.458249	0.448621	0.476696
Rocchio-2	0.495	0.444119	0.454437	0.448007	0.474435
Rocchio-3	0.505	0.455495	0.463649	0.449008	0.485906
Rocchio-4	0.497	0.450539	0.460866	0.448778	0.483619
Rocchio-5	0.497	0.441519	0.449421	0.441622	0.472068
Rocchio-6	0.479	0.428432	0.443400	0.439774	0.466213
Avg	0.500	0.444096	0.455004	0.445968	0.476490

For each topic, the system starts from an initial training set, then adds a window of new training documents. The size of the window set is 25. Each window of new training documents is selected randomly in the testing set. To test the robustness of the proposed model, we conduct the process of adaptive six times (six windows) for the same initial training set. Table 7 shows the results of the *N-PTM* models which combine new training documents with the initial one into a big training set and then train the system again.

We also test the adaptive performance of the term-based baseline models for the same settings, and found that the Rocchio model achieves the best performance. Table 8 shows the results of adaptive Rocchio models for using the six windows. The experiment results show that the adaptive N-PTM models also achieve excellent performance with 9.10% (max 12.00% and min 6.28%) average percentage change for all five measures on the assessor topics. We believe that the performance of N-PTM model is consistent and very significant for all five measures on the RCV1 data collection.

6.4 Discussion

The main process of the proposed approach consists of two steps: offender selection, and the revision of term weights. It is obvious that not all negative documents are suitable to be selected as offenders, where offenders are the most useful negative documents that can help to balance the percentages of general terms and specific terms in the extracted features. Informally, the documents that have high weight are called offenders.

Table 9 shows the statistical information for N-PTM with different values of K including the average numbers of offenders, extracted terms and their weights, and the performance. The results of *11-points* on all assessor topics for the different values of K are

Table 9 Statistical information for N-PTM with different values of K

K	Number of offenders	Number of extracted terms			Average weight of extracted term			*Top-20*	*MAP*	$F_{\beta=1}$
		$\#T^+$	$\#T^-$	$\#G$	$w(T^+)$	$w(T^-)$	$w(G)$			
$\lvert D^+\rvert/3$	6.5	107	126	50	36.33	−26.54	56.04	0.547	0.486	0.463
$\lvert D^-\rvert$	26.5	86	343	71	39.21	−319.3	73.41	0.442	0.372	0.375
$\lvert D^+\rvert/2$	9.7	100	145	57	33.31	−46.11	58.56	0.538	0.479	0.458
$\lvert D^+\rvert$	16.5	93	208	64	33.68	−96.09	61.79	0.538	0.452	0.441

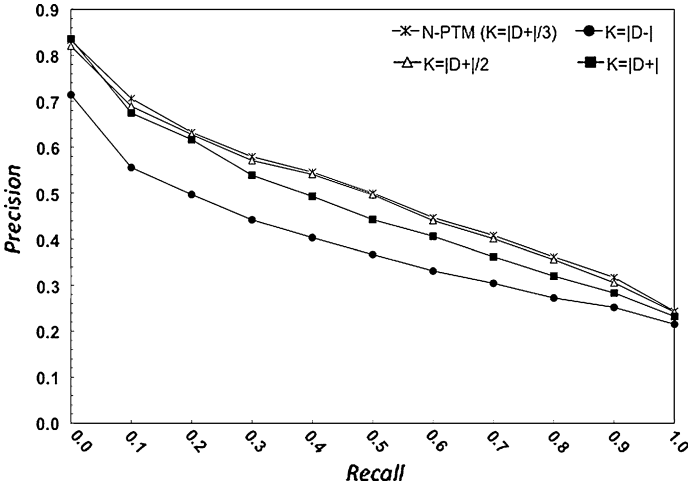

Fig. 2 Comparison between used all negative documents and used the offender one

Table 10 Performance of each loop for all assessor topics

Loop	*Top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
Loop 1	0.5240	0.4560773	0.4667359	0.4522290	0.4884621
Loop 2	0.5500	0.4619442	0.4815856	0.4601250	0.5012464
Loop 3	0.5470	0.4718463	0.4863368	0.4631121	0.5066751
Loop 4	0.5470	0.4561296	0.4817219	0.4595607	0.5015123
Loop 5	0.5440	0.4521346	0.4771687	0.4573536	0.497056
Loop 6	0.5510	0.4502285	0.4714463	0.4537675	0.4927944

reported in Fig. 2. It is obvious that $K = \lceil \frac{|D^+|}{3} \rceil$ is the best one. The statistical information illustrates that the proposed method for offender selection meets the design objectives.

As mentioned in Sect. 5.2, we used three loops to get negative specific terms in order to reduce the side effects of using general terms. Table 10 illustrates the performance of the loops used in Algorithm *NFMining* (up to 6 loops). The table shows that the system achieves the best result in average after the third loop.

Table 11 shows the average numbers of positive documents, negative documents, offenders and extracted terms in the training sets for the loops in the algorithm *SPMining*. Based on the proposed model, we set $K = \lceil \frac{|D^+|}{3} \rceil$, that is, the number of offender documents

Table 11 Extracted features in the loops of algorithm *SPMining*,where $min\_sup = 0.2$

Loop	Number of documents in training sets				Number of extracted terms			Weight of extracted terms		
	Positive	Negative	Offenders in the memory	Offenders used in the loop	$\#T^+$	$\#T^-$	$\#GT$	$w(T^+)$	$w(T^-)$	$w(GT)$
Loop 1	12.8	41.3	3.98	3.98	122	59	35	37.59	−9.68	48.81
Loop 2	12.8	41.3	5.54	3.98	110	110	47	31.88	−18.48	54.51
Loop 3	12.8	41.3	6.5	3.98	107	126	50	30.36	−26.54	56.04
System	*12.8*	*41.3*	*6.5*	-	*107*	*126*	*50*	*36.33*	*−26.54*	*56.04*
Loop 4	12.8	41.3	7.44	3.98	104	140	53	29.50	−35.84	56.89
Loop 5	12.8	41.3	8.12	3.26	103	147	54	29.14	−42.70	57.26
Loop 6	12.6	41.0	8.86	3.18	100	152	55	28.26	−48.52	57.40

should be equal or less than the number of positive documents. We also use loops to calculate the closeness of the offender document to the positive, such as, if a document is very closed to the positive documents it will be ranked at the top for the next loop. As shown in Table 11 the average number of positive documents is about 13 and the average number of negative documents is about 41; however, the average number of offender documents that have been selected in each loop is only 4 or 3. The table also illustrates that only $15.74\% = \frac{6.5}{41.3}$ negative documents are selected as offenders for the system, that is, the proposed method is much efficient for reducing the space of negative documents.

For the revision of term weights, the proposed method first classifies extracted terms into general terms and specific terms that is a distinguish advantage comparing with others [35, 46]. The normal belief is that specific terms are more interesting than general terms for a given topic. Therefore, the proposed method increases the weights of positive specific terms when it conduces the revision using negative documents.

General terms are not only frequently appear in positive documents, but also frequently appear in some negative documents because negative documents may describe some extent to which the topic discusses what users want. To reduce the side effects of using general terms in the extracted features, the proposed method adds negative specific terms (and negative weights) into the extracted features by the loops (see Algorithm *NFMining*).

Table 11 also shows the average numbers of extracted general terms $\#GT$, specific terms $\#T^+$ and negative specific terms $\#T^-$, and their average weights. For the system, before revision, it can be seen that more than $61\% = \frac{56.04}{56.04+36.33}$ weights are distributed to general terms although the percentage of general terms is $31.8\% = \frac{50}{50+107}$ for all extracted terms in positive documents.

After revision, 126 negative specific terms are added into T in average for the system (see Table 11), and they are assigned weight −26.54 in average. In this way, these negative specific terms could reduce the side effects of general terms if both general terms and negative specific terms appear in negative documents because now only $45\% = \frac{56.04-26.54}{56.04-26.54+36.33}$ weights could be distributed to general terms considering positive specific terms get weight 36.33 in average and general terms get $29.5 = 56.04 - 26.54$ in average.

The above analysis illustrates that the proposed algorithm for finding negative specific terms meets the design objective for reducing the side effects of using general terms.

7 Conclusions

Negative relevance feedback is very useful for information filtering. However, whether negative feedback can largely improve filtering accuracy is still an open question. This paper presents a pattern mining based approach for this open question. It introduces a method to select negative documents (or called offenders) that are close to the extracted features in the positive documents. It also proposes an approach to classify extracted terms into three groups: positive specific terms, general terms and negative specific terms. In this perspective, it presents an iterative algorithm to revise extracted features. Compared with the state-of-the-art models, the results of experiments on the RCV1 data collection demonstrate that the effectiveness of information filtering can be significantly improved by the proposed new approach, and the performance is also consistent for adaptive filtering. This research provides a promising methodology for evaluating term weights based on discovered patterns (rather than documents) in both positive and negative relevance feedback.

Acknowledgements This paper was partially supported by an ARC Discovery Grant from Australian Research Council (Project ID: DP0988007). We also would like to thank Prof. Peter Bruza and Prof. Ning Zhong for their constructive comments.

References

1. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of 27th international conference on very large databases (VLDB'01)*, (pp. 478–499).
2. Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Reading: Addison Wesley.
3. Belkin, N. J., & Croft, W. B. (1992). Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM, 35*(12), 29–38.
4. Fu, X., Budzik, J., & Hammond, K. J. (2000). Mining navigation history for recommendation. In *Proceedings of the 5th international conference on Intelligent user interfaces (IUI'00)*, (pp. 106–112).
5. Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of 2000 ACM SIGMOD international conference on management of data (SIGMOD'00)*, (pp. 1–12).
6. Iwayama, M. (2000). Relevance feedback with a small number of relevance judgements: Incremental relevance feedback vs. document clusting. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'00)*, (pp. 10–16).
7. Jindal, N., & Liu, B. (2006). Identifying comparative sentences in text documents. In *Proceedings of SIGIR'06*, (pp. 244–251).
8. Jones, K. S., Walker, S., & Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments—part 1. *Information Processing and Management, 36*(6), 779–808.
9. Lau, R. Y. K., Bruza, P., & Song, D. (2004). Belief revision for adaptive information retrieval. In *Proceedings of SIGIR'04*, (pp. 130–137).
10. Lavrenko, V., & Croft, W. (2001). Relevance-based language models. In *Proceedings of SIGIR'01*, (pp. 120–127).
11. Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR'92*, (pp. 37–50).
12. Li, X. & Liu, B. (2003). Learning to classify texts using positive and unlabeled data. In *Proceedings of international joint conference on artificial intelligence (IJCAI'03)*, (pp. 587–594).
13. Li, Y., Algarni, A., Wu, S.-T., & Xu, Y. (2009). Mining negative relevance feedback for information filtering. In *Proceedings of 2009 IEEE/WIC/ACM international conference on web intelligence*, (pp. 606–613).
14. Li, Y., & Zhong, N. (2006). Mining ontology for automatically acquiring web user information needs. *IEEE Transactions on Knowledge and Data Engineering, 18*(4), 554–568.
15. Li, Y., Zhou, X., Bruza, P., Xu, Y., & Lau, R. Y. (2008). A two-stage text mining model for information filtering. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM'08)*, Napa Valley, California, USA, (pp. 1023–1032).

16. Liu, B. (2007). *Web data mining: Exploring hyperlinks, contents, and usage data (Data-Centric Systems and Applications)*. Springer, January 2007.
17. Lv, Y., & Zhai, C. (2009). Adaptive relevance feedback in information retrieval. In *Proceedings of CIKM'09*, (pp. 255–264).
18. Metzler, D., & Croft, W. (2007). Latent concept expansion using markov random fields. In *Proceedings of SIGIR'07*, New York, NY, USA, ACM.
19. Mostafa, J., & Lam, W. (2000). Automatic classification using supervised learning in a medical document filtering application. *Information Processing and Management, 36*(3), 415–444.
20. Mostafa, J., Mukhopadhyay, S., Lam, W., & Palakal, M. J. (1997). A multilevel approach to intelligent information filtering: Model, system, and evaluation. *ACM Transactions on Information Systems, 15*(4), 368–399.
21. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of 17th international conference on data engineering (ICDE'01)*, (pp. 215–224).
22. Qin, T., Zhang, X.-D., Wang, D.-S., Liu, T.-Y., Lai, W., & Li, H. (2007). Ranking with multiple hyperplanes. In *Proceedings of SIGIR'07*, (pp. 279–286).
23. Robertson, S. E., & Soboroff, I. (2002). The trec 2002 filtering track report. In *Proceedings of TREC'02*.
24. Robertson, S. E., Walker, S., & Hancock-Beaulieu, M. (1998). Okapi at trec-7: Automatic ad hoc, filtering, vlc and interactive. In *Proceedings of TREC'98*, (pp. 199–210).
25. Robertson, S. E., Zaragoza, H., & Taylor, M. J. (2004). Simple bm25 extension to multiple weighted fields. In *Proceedings of CIKM'04*, (pp. 42–49).
26. Rocchio, J. (1971). *Relevance feedback in information retrieval*, volume In the SMART retrieval system: Experiments in automatic document processing. Prentice Hall.
27. Scott, S., & Matwin, S. (1999). Feature engineering for text classification. In *Proceedings of 16th international conference on machine learning*, 1999. Scott, Sam and Matwin, Stan, (pp. 379–388).
28. Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*(1), 1–47.
29. Shen, X., Tan, B., & Zhai, C. (2005). Context-sentitive information retrieval using implicit feedback. In *Proceedings of SIGIR'05*, (pp. 43–50).
30. Soboroff, I. & Robertson, S. (2003). Building a filtering test collection for trec 2002. In *Proceedings of SIGIR'03*, (pp. 243–250).
31. Song, F., & Croft, W. (1999). A general language model for information retrieval. In *Proceedings of CIKM'99*, (pp. 316–321).
32. Wu, S. T., Li, Y., Xu, Y., Pham, B., & Chen, P. (2004). Automatic pattern-taxonomy extraction for web mining. In *Proceedings of 2004 IEEE/WIC/ACM international conference on web Intelligence*, pp 242–248, China.
33. Turmo, J., Ageno, A., & Catal, N. (2006). Adaptive information extraction. *ACM Computing Surveys, 38*(2): (Article No. 4).
34. Wang, X., Fang, H., & Zhai, C. (2007). Improve retrieval accuracy for difficult quries using negative feedback. In *Proceedings of CIKM'07*, (pp. 991–994, pp. 991–994)
35. Wang, X., Fang, H., & Zhai, C. (2008). A study of methods for negative relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'08)*, (pp. 219–226), New York, NY, USA, 2008. ACM.
36. Wu, S.-T. (2007). *Knowledge discovery using pattern taxonomy model in text mining*. PhD Thesis, Queensland University of Technology.
37. Wu, S.-T., Li, Y., & Xu, Y. (2006). Deploying approaches for pattern refinement in text mining. In *Proceedings of ICDM'06*, (pp. 1157–1161).
38. Xu, J., & Croft, W. (1996). Query expansion using local and global doscument analysis. In *Proceedings of SIGIR'96*, New York, NY, USA. ACM, (pp. 4–11).
39. Xu, J., & Croft, W. (2000). Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems, 18*(1), 79–112.
40. Xu, Y., & Li, Y. (2007). Generating concise association rules. In *Proceedings of CIKM'07*, (pp. 781–790).
41. Yan, X., Cheng, H., Han, J., & Xin, D. (2005). Summarizing itemset patterns: A profile-based approach. In *Proceedings of 11th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'05)*, (pp. 314–323).
42. Yang, Y., Lad, A., Lao, N., Harpale, A., Kisiel, B., & Rogati, M. (2007). Utility-based information distillation over temporally sequenced documents. In *Proceedings of SIGIR'07*, (pp. 31–38).
43. Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of SIGIR'99*, (pp. 42–49).

44. Yang, Y., Yoo, S., Zhang, J., & Kisiel, B. (2005). Robustness of adaptive filtering methods in a cross-benchmark evaluation. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR05)*, (pp. 98–105).
45. Zhai, C., & Lafferty, J. (2001). Model-based feedback in language modeling approach to information retrieval. In *Proceedings of CIKM'01*, (pp. 403–410).
46. Zhang, Y. (2004). Using bayesian priors to combine classifiers for adaptive filtering. In *Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'04)*, (pp. 345–352).
47. Zhang, Y., & Callan, J. (2005). Combining multiple forms of evidence while filtering. In *Proceedings of the conference on human language technology and empirical methods in natural language processing (HLT'05)*, Morristown, NJ, USA, (pp. 587–595).