Relevance Feature Discovery for Text Analysis

by

Abdulmohsen Algarni

A thesis submitted for the degree of

Doctor of Philosophy

Faculty of Science and Technology

Queensland University of Technology

October 2011

To my family with love and respect...

Keywords

Feature selection, Pattern Taxonomy Model, Information Retrieval, Text Mining, Data Mining, Association Rules, Sequential Pattern Mining, Closed Sequential Patterns, Pattern Deploying, Pattern Evolving, Offender selection, Weight revision.

QUEENSLAND UNIVERSITY OF TECHNOLOGY

Abstract

Faculty of Science and Technology

Doctor of Philosophy

by

Abdulmohsen Algarni

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. This research presents a promising method, Relevance Feature Discovery (RFD), for solving this challenging issue. It discovers both positive and negative patterns in text documents as high-level features in order to accurately weight low-level features (terms) based on their specificity and their distributions in the high-level features.

The thesis also introduces an adaptive model (called ARFD) to enhance the flexibility of using RFD in adaptive environment. ARFD automatically updates the system's knowledge based on a sliding window over new incoming feedback documents. It can efficiently decide which incoming documents can bring in

new knowledge into the system. Substantial experiments using the proposed models on Reuters Corpus Volume 1 and TREC topics show that the proposed models significantly outperform both the state-of-the-art term-based methods underpinned by Okapi BM25, Rocchio or Support Vector Machine and other pattern-based methods.

Contents

List of Figures

List of Tables

Declaration of Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed: _____

Date: _____

Acknowledgements

This research project would not have been possible without the support of many people. First of all, I would like to express my immense gratitude to Professor Yuefeng Li, my principal supervisor, for all his guidance and encouragement throughout this research work. He has been always there providing sufficient support with his excellent expertise in this area. Many thanks also go to my associate supervisor, Dr. Yue Xu for her generous support and comments on my work during this candidature.

This work would not have been accomplished without the constant support of my family. I would like to dedicate this thesis to my family for their never-ending encouragement over these years. Special thanks go to my wife for listening and supporting me in the last couple of years. I am indebted to my many colleagues who have supported me especially group members; Dr. Xiaohui Tao and Dr. Huizhi Liang for offering invaluable advice and comments regarding my research work. Thanks also go to Dr. Sheng-Tang Wu (Sam) for the data pre-processing and the construction of some baseline models. Special thanks must go to the Faculty of Science and Information Technology, QUT, which has provided me the comfortable research environment with needed facilities and financial support including travel allowances over the period of my candidature. Also many thanks to the proofreader for their generous work in my thesis. Finally, I would also like to thank my examiners for their precious comments and suggestions.

Chapter 1

Introduction

1.1 Motivation

Search engines retrieve a list of sometimes thousands or millions of pages based on a submitted query regardless of who submits it [82]. For example, when the same query is submitted by different users, most search engines return the same results to all the different users [82]. Usually, most of the retrieved documents are irrelevant to what the user needs. For example, for the query "Java", some users may be interested in documents about "Java programming language", while other users may want documents about "Coffee". It has become an essential to provide users with effective tools that help to remove most of the redundant and unwanted documents.

Generally, Information Filtering (IF) can provide efficient tools to help people find the most valuable information, in order to limit time spent on searching. Information filtering can be categorized into three categories: adaptive filtering,

batch filtering, and routing filtering. In adaptive filtering, the system starts with a user profile or a limited number of feedback documents to build a user profile and immediately begin filtering documents. In this simulation, the system can use the provided new feedback documents to adaptively update the user profile. The provided documents can be a pseudo feedback, which is the top relevance documents judged by the system or relevance feedback documents judged by the user.

Many models have been developed to deal with adaptive issues in the IF area. Most of these models focus on updating the model parameters to calibration and optimising thresholds and to follow the changes in the user's interest. These problems have been studied in the adaptive filtering area and include topic profile adaptation using incremental Rocchio, Gaussian-Exponential density models, logistic regression in a Bayesian framework, etc; and threshold optimisation strategies using probabilistic calibration or local fitting techniques [10, 73, 100, 101, 105, 106].

Batch filtering is identical to adaptive filtering, except the system also starts with a large sample of evaluated feedback documents. This makes it much more like a text categorisation task [75]. Routing is very similar to batch filtering, but in this case, the system is expected to return a ranked list of documents which will be evaluated according to traditional information retrieval (IR) methods. Routing filtering systems return a list of ranked documents rather than making boolean decisions.

Relevance feedback has been used widely in the area of information filtering and information retrieval. It has been shown to be effective with different kinds of

retrieval models [34, 66, 69, 72, 104]. The objective of using relevance feedback is to find useful features available in a feedback set, including both positive and negative documents, for describing what users need. However, 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. This is a particularly challenging task in modern information analysis, from both an empirical and a theoretical perspective [44, 47]. This problem is also of central interest in many web-personalised applications, and has received attention from researchers in Data Mining, Web Intelligence and Information Retrieval (IR) communities.

Many IR models have been developed for relevance feedback to solve this challenge [52, 65]. There are two major classes in IR history: global methods and local methods [51, 92], 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 [29, 69], Probabilistic models and Okapi BM25 [30, 62], and language models, including model-based methods and relevance models [34, 52, 58, 88, 104]. In a language model, the key elements are the probabilities of word sequences that include both words and phrases (or sentences). They are often approximated by *n-gram* models [80], such as Unigram, Bigram or Trigram, for considering term dependencies.

Most models in IR are term-based methods [52, 65]. The popular Rocchio model is one of the effective term-based models that utilises relevance feedback to build the user profile. It uses both positive and negative feedback to update user profiles

in vector space models. It can be generalised as follows: [88, 105]

$$U = \alpha \frac{1}{|D^+|} \sum_{\overrightarrow{d} \in D^+} \frac{\overrightarrow{d}}{|| \overrightarrow{d} ||} - \beta \frac{1}{|D^-|} \sum_{\overrightarrow{d} \in D^-} \frac{\overrightarrow{d}}{|| \overrightarrow{d} ||} \tag{1.1}$$

where U is the user profile vector, D^+ is the set of relevant documents, D^- is the set of non-relevant documents, and α and β are the empirical parameters.

Rocchio model Eq. (1.1) has two parts. The first part of the equation uses important terms that appear in positive documents. The second part of the equation uses negative terms extracted from negative feedback documents to reduce the side effects of using the important terms.

The advantages of term-based methods include efficient computational performance; as well as mature theories for term weighting. However, terms suffer from the problems of synonymy and polysemy. A synonymy is a word, which shares the same meaning as another word (e.g. taxi and cab). Polysemy is a word that has two or more meanings (e.g. river "bank" and CITI "bank"). For the Rocchio model, it is hard to understand what terms are affected by these two problems if the performance is poor.

In addition, using phrases to represent documents and for queries has generally been accepted as a desirable feature in information retrieval. A phrase refers to the concatenation of two or more words which must occur in text separated only by white space and does not range across paragraph or sentence bounds [79]. Nonetheless, these phrase-based methods did not yield significant improve the performance because phrases with high frequency (normally shorter phrases) usually have a high value on exhaustivity but a low value on specificity, and thus

specific phrases encounter the low frequency problem [89].

Naturally, phrases have inferior statistical properties to words, and there are large numbers of redundant and noisy phrases among extracted phrases in the documents [74, 75]. Therefore, it is challenging to find only the useful phrases for text mining and classification. From the perspective of data mining, some researchers thought sequential patterns could be an alternative to phrases [28, 90]. Sequential patterns in a text document refer to a list of terms that appear together in a sentence, paragraph or document in the same order.

Pattern mining has been extensively studied in data mining communities for many years. Many efficient algorithms such as Apriori-like algorithms, PrefixSpan, FP-tree, SPADE, SLPMiner and GST [24, 26, 76, 97, 103] have been proposed. Two kinds of patterns can be extracted from text documents: Intra-patterns and Inter-patterns. Intra-patterns are patterns that appear in sentences. On the other hand, Inter-patterns are patterns that appear across sentences in paragraphs. A sequential pattern is called a frequent pattern if its frequency is greater than a threshold ($minimum\_support$). However, a lot of patterns can be extracted from documents, and most of them are either redundant or noisy patterns. Currently, data mining has developed some techniques (e.g. maximal patterns, closed patterns and master patterns) for removing redundant and some noisy patterns [93, 96, 97]. The following example explains the difference between frequent patterns and closed sequential patterns.

Figure 1.1 shows a sample of feedback documents from the RCV1 dataset. Frequent patterns are extracted from that sample document using the PrefixSpan

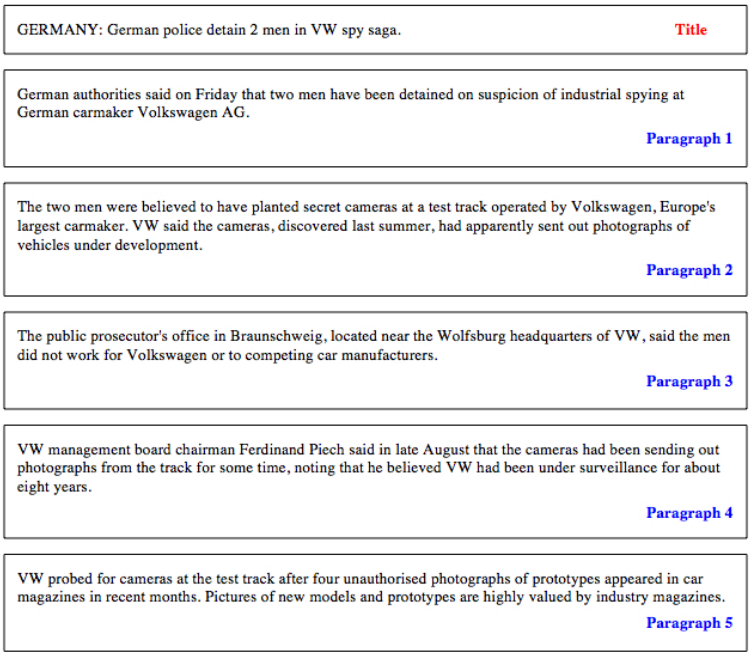

FIGURE 1.1: A sample of a document in a training set.

algorithm [48]. These patterns also include some special patterns, where a special pattern consists of a single term. As shown in Table 1.1 about 57 frequent patterns with $minimum\_sup$ $(\xi) \geq 0.2$ have been extracted from the sample document. Closed patterns technique can be used to remove redundant patterns and most of the noisy patterns. A frequent pattern P is a closed pattern if there exists no frequent patterns P' such that $P \sqsubset P'$ and $frequency(P) = frequency(P')$. In this example only 16 patterns are closed sequential patterns out of 57 frequent patterns. A list of closed sequential patterns extracted from the sample document is shown in Table 1.2.

Closed patterns used in data mining community have turned out to be an alternative to phrases [28, 91] because patterns enjoy good statistical properties like terms.

TABLE 1.1: Frequent pattern extracted from the sample training denouement in Figure 1.1 ($minimum\_sup$ (ξ) ≥ 0.2).

Frequent Pattern	Frequency	Frequent Pattern	Frequency
$\langle test \rangle$	2	$\langle german, detain \rangle$	2
$\langle vw \rangle$	5	$\langle german, men \rangle$	2
$\langle men \rangle$	5	$\langle german, spy \rangle$	2
$\langle camera \rangle$	3	$\langle vw, camera \rangle$	3
$\langle track \rangle$	3	$\langle vw, track \rangle$	2
$\langle photograph \rangle$	3	$\langle vw, photograph \rangle$	3
$\langle industri \rangle$	2	$\langle vw, car \rangle$	2
$\langle work \rangle$	2	$\langle men, spy \rangle$	2
$\langle car \rangle$	3	$\langle men, volkswagen \rangle$	4
$\langle spy \rangle$	2	$\langle men, car \rangle$	2
$\langle german \rangle$	2	$\langle men, vw \rangle$	2
$\langle believ \rangle$	2	$\langle men, work \rangle$	2
$\langle detain \rangle$	2	$\langle men, carmak \rangle$	2
$\langle carmak \rangle$	2	$\langle vw, camera, photograph \rangle$	3
$\langle prosecutor \rangle$	2	$\langle vw, camera, track \rangle$	2
$\langle test, photograph \rangle$	2	$\langle camera, track, photograph \rangle$	2
$\langle test, track \rangle$	2	$\langle camera, track, vw \rangle$	2
$\langle believ, vw \rangle$	2	$\langle men, work, car \rangle$	2
$\langle detain, spy \rangle$	2	$\langle men, work, volkswagen \rangle$	2
$\langle camera, photograph \rangle$	3	$\langle men, volkswagen, car \rangle$	2
$\langle camera, vw \rangle$	2	$\langle work, volkswagen, car \rangle$	2
$\langle camera, track \rangle$	3	$\langle test, track, photograph \rangle$	2
$\langle camera, test \rangle$	2	$\langle camera, test, photograph \rangle$	2
$\langle work, car \rangle$	2	$\langle camera, test, track \rangle$	2
$\langle work, volkswagen \rangle$	2	$\langle german, detain, spy \rangle$	2
$\langle track, photograph \rangle$	2	$\langle german, men, spy \rangle$	2
$\langle track, vw \rangle$	2	$\langle camera, test, track, photograph \rangle$	2
$\langle volkswagen, car \rangle$	2	$\langle men, work, volkswagen, car \rangle$	2

Pattern taxonomy model (PTM) [45, 90, 91] have been proposed for using closed sequential patterns in text classification. To effectively use closed sequential patterns in a PTM, a deploying method has been proposed to compose all closed sequential patterns from a category into a vector that included a set of terms and a term-weight distribution [41]. An example of deploying patterns over terms is shown in Figure 1.2, where terms in the pattern are considered equally. These

TABLE 1.2: Closed sequential patterns patterns extracted from the sample training denouement in Figure 1.1

Closed Sequential Patterns	Frequency
$\langle vw \rangle$	5
$\langle men \rangle$	4
$\langle industri \rangle$	2
$\langle vw, car \rangle$	2
$\langle believ, vw \rangle$	2
$\langle men, volkswagen \rangle$	3
$\langle men, vw \rangle$	2
$\langle men, carmak \rangle$	2
$\langle camera, photograph \rangle$	3
$\langle camera, track \rangle$	3
$\langle german, men, spy \rangle$	2
$\langle vw, camera, photograph \rangle$	3
$\langle vw, camera, track \rangle$	2
$\langle german, detain, spy \rangle$	2
$\langle camera, track, vw \rangle$	2
$\langle camera, test, track, photograph \rangle$	2

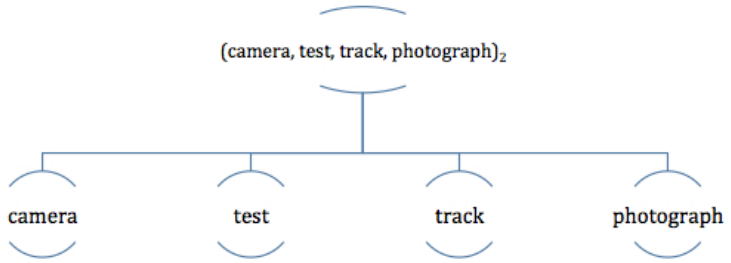

FIGURE 1.2: Deploying patterns to a term space model.

pattern mining based approaches have shown a certain extent of improvement over the effectiveness of using only positive feedback. However, less significant improvements are made compared with term-based methods. The most likely reason is that many patterns may contain noise when extracted from the positive document only. Furthermore, PTM is lacking in their ability to effectively use negative feedback documents.

Many people believe that there is plenty of negative information available and negative documents are very useful because they can help users to search for accurate information [88]. Existing methods which use both positive and negative feedback for IF can be group into two approaches. The first approach is to revise terms that appear in both positive and negative samples. For instant, Rocchio-based models and SVM [65] based filtering models. 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 samples and negative samples. For example, probabilistic models [7], and BM25 [65]. In a pattern-based approach, using negative feedback is a challenging issue. Nevertheless, using negative feedback in pattern-based approaches could improve the quality of extracted features by reducing noises features.

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 [57] and SVM [17, 60]). 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 [94] or BM25 [64, 65]). The third one is to describe features as positive patterns [90, 91]. In this thesis, we further develop the third approach by effectively using both high-level patterns and low-level terms extracted from both positive and negative patterns.

1.2 Problem Statement and Our Solution

To overcome the limitation of term-based approaches, sequential patterns and closed sequential patterns have been developed in pattern taxonomy models (PTM) [90, 91]. This approach introduced data mining techniques to information filtering. The PTM has shown a certain extent of improvement in effectiveness using only positive feedback; however, too many noisy features extracted from positive feedback adversely affect PTM systems [45]. We believe that using negative feedback can lead to a significant improvement for extracting features from positive feedback documents in the PTM model. Zhong, et al. recently proposed the use of negative feedback in the PTM model in [56]. The result shows that using both positive and negative feedback can give better effectiveness results than using positive feedback only. However, the improvement gain from using both positive and negative feedback is not significant compared to using only positive feedback. Therefore, whether negative feedback can indeed largely improve filtering accuracy is still an open question for pattern-based approaches.

To solve this question, we analysed the PTM and have made the following observations: First, there is a large amount of negative feedback, most of which is not useful and can be eliminated by using positive feedback. Second, discovered features (especially terms) have a different focus depending on the given topic. Third, features extracted from both positive and negative documents include noise. Based on these observations, we introduce a relevance feature discovery model (RFD).

Formally, let D be a set of training documents, which consists of positive feedback D^+ and negative feedback D^-. One of the objectives of Relevance Feature

Discovery is to extract low-level features (or called terms) from D and group them into three categories: the positive specific terms S^+ which describe the specific thinks about what users need, general terms G which describe some related things to what users need, and the negative specific features S^- which describes some things that are not what users want. Therefore, the key research questions are how to find a set of useful terms and how to effectively classify the useful terms into the three categories.

The RFD model uses negative feedback documents to improve the quality of extracted features from positive feedback documents in the PTM model. The RFD model utilises two levels of features, high-level patterns (closed sequential patterns) and low-level terms, to build user profiles. The high-level patterns are used to improve the feature selection task. Then, to overcome the disadvantages of low frequency patterns, the high-level patterns are deployed into low-level terms. In order to reduce the space of negative feedback, some of negative feedback (called offenders) are selected, where an offender is a negative document that is closed to positive documents. Moreover, we introduce a specificity function to calculate the specificity of low-level terms to understand what the topic is focused on. Then, low-level terms can be categorised into three categories based on the specificity function: specific positive, general and specific negative. In that way, multi-updating methods can be used to revise the weights of low-level terms.

To take advantage of new feedback documents, the RFD model has been extended to be an adaptive model (ARFD). The new ARFD model evaluates the new feedback documents using the knowledge (called base knowledge) that the system has. The documents that are correctly classified by the system will be discarded because they are redundant documents. Then, new knowledge can be extracted

from the selected documents in the new training dataset. A merging function has been developed to merge both base knowledge and the new knowledge. The goal of the adaptive model is to update the system efficiently with new knowledge to improve the effectiveness.

1.3 Contributions

There are two models proposed in this thesis: Relevance Feature Discovery (RFD) and Adaptive Relevance Feature Discovery (ARFD). We list our contributions to each of them below:

- **Relevance Feature Discovery.**
    1. We present a Relevance Feature Discovery model (RFD) to integrate both high-level patterns and low-level terms.
    2. We introduce a strategy to select constructive negative samples (offenders) in order to reduce the space of negative documents.
    3. We also define a specificity function to classify low-level terms into three groups: Specific positive group, General group, and Specific negative group.
    4. We provide a promising methodology for evaluating term weights based on both their distribution in patterns and their specificity scores.
- **Adaptive Relevance Feature Discovery.**
    1. We present an Adaptive Relevance Feature Discovery model (ARFD), based on the relevance feature discovery model.

2. We introduced a strategy to evaluate new feedback documents and select only documents that contain knowledge new to the system.

3. We propose a new method to integrate the new knowledge with the existing knowledge in the system (base knowledge).

1.4 Publications

- Abdulmohsen Algarni, Yuefeng Li, and Yue Xu. Adaptive Information Filtering Based on PTM Model (APTM). In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03 (WI-IAT '08), Vol. 3.*, pages 37-40, Washington, DC, USA.

- Yuefeng Li, Abdulmohsen Algarni, S.-T. Wu, and Yue Xu. Mining negative relevance feedback for information filtering. In *WI-IAT '09: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 606–613, Washington, DC, USA, 2009. IEEE Computer Society.

- Abdulmohsen Algarni, Yuefeng Li, Yue Xu, and R. Y. Lau. An effective model of using negative relevance feedback for information filtering. In *Proceeding of the 18th ACM conference on Information and Knowledge Management (CIKM '09),*, pages 1605–1608, New York, NY, USA, 2009. ACM.

- Yuefeng Li, Xiaohui Tao, Abdulmohsen Algarni, and Sheng-Tang Wu. Mining Specific and General Features in Both Positive and Negative Relevance

Feedback. In *Proceedings of the 18th Text Retrieval Conference (TREC 2009)*, 2009.

- Abdulmohsen Algarni, Yuefeng Li, and Yue Xu. Selected new training documents to update user profile. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10).*, pages 799-808, ACM, New York, NY, USA.

- Yuefeng Li, Abdulmohsen Algarni, and N. Zhong. Mining positive and negative patterns for relevance feature discovery. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*, pages 753-762, New York, NY, USA, 2010.

- Abdulmohsen Algarni, Yuefeng Li, and Xiaohui Tao. Mining Specific and General Features in Both Positive and Negative Relevance Feedback. In *Proceedings of the 19th Text Retrieval Conference (TREC 2010)*, 2010.

- Yuefeng Li, Abdulmohsen Algarni, and Yue Xu. A pattern mining approach for information filtering systems. *Information Retrieval*,Vol.14, No.3, pages 237–256, 2011.

- Abdulmohsen Algarni, Yuefeng Li, Sheng-Tang Wu, and Yue Xu. Text mining in negative relevance feedback. Accepted in *Web Intelligence and Agent Systems, An International Journal.*

1.5 Thesis Structure

The rest of this thesis is summarised as follows:

Chapter 2: This chapter is a literature review of related disciplines including data mining, text mining, knowledge representation models and information filtering. It reviews the current work on data mining and identifies the drawbacks of existing representation schemes.

Chapter 3: This chapter provides definitions and a theoretical framework for the proposed models. It also introduces some of the proposed model's related work. This chapter also presents a novel representation scheme that makes use of the discovered pattern taxonomies.

Chapter 4: This chapter describes the RFD model presented in chapter 3 in more detail. Two algorithms are introduced in this chapter. The first is for extracting high-level patterns and low-level terms. The second algorithm uses negative feedback to revise the weight of low-level terms.

Chapter 5: This chapter presents the adaptive model for RFD. The proposed algorithm of the ARFD model is discussed in this chapter.

Chapter 6: This chapter gives the description of benchmark datasets and performance measures, along with the application of the proposed models to the information filtering. A detailed analysis of the comparison results of experiments performed is also presented in this chapter.

Chapter 7: This chapter concludes this thesis and outlines the direction for future work.

Chapter 2

Literature Review

A literature review is an evaluation of the information found in the literature related to a particular area of study. In order to utilise these benefits, this literature review examines previous research on related topics.

2.1 Introduction

Every day, Web resources increase by a large number of new documents. Web users need effective tools that enable them to find the information they need quickly. This is a very challenging problem because user needs change over time and there is a limited amount of available feedback data. Building an information filtering model that matches user needs to user profiles is a complex challenge. This research will examine this problem from different perspectives, as shown in Figure 2.1. The figure shows the main areas of the this research that need to be studied.

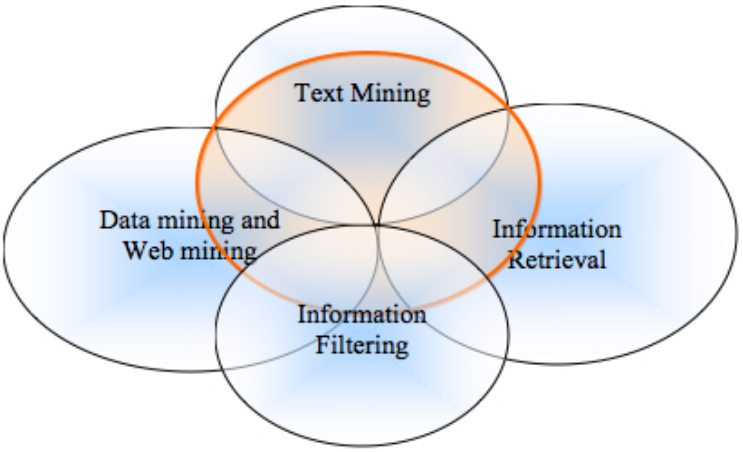

FIGURE 2.1: Research areas.

This literature review examines previous research on related topics. Specifically, the literature will examine research on data mining, information retrieval, information filtering, and text mining which are the main topics related to this research.

2.2 Association Rules and Sequential Patterns

Association rule mining represents the fundamental data mining task of analysing data to discover what elements frequently co-occur in a dataset containing multiple independent selections. The traditional application of association rule mining is market basket data analysis. In supermarket basket data, the user can figure out which items are being sold most frequently.

$$Cheese \rightarrow Beer \qquad [support = 10\%,\ confidence = 80\%]$$

The rule reveals that 10% of customers buy cheese and beer together and that most of the time, about 80% of customers who buy cheese also buy beer. However, this is not the only type of trend that can be identified. The goal of database mining is to automate the process of finding interesting patterns and trends. Once this information is available, we can perhaps omit the original database. The main output of the data mining process is a summary of the database. Association rule mining is very popular in data mining and can be used for many applications. In Web and text mining, association rule mining is utilised to find word co-occurrences.

Association rules mining does not consider the sequence of the items that co-occur frequently in a dataset. However, sequential pattern mining takes care of that. An example of sequential pattern mining can be described as follows: 5% of customers buy beds first, then mattresses, and then pillows. The items are not purchased at the same time, but one after another.

2.2.1 Association Rule Approach

The association rule approach has two phases. The first is the support phase, where frequent itemsets are identified, and the second phase is confidence, where conditional probabilities are identified in transactions where items appear together continually. One practical application of association analysis is market basket analysis. For example, a supermarket manager wishes to boost sales without increasing expenditure. One way to accomplish this is to change the strategic position of products to entice consumers to spend more. The problem of mining association rules from large data sets should be addressed as follows:

- Find all items whose support is greater than the user's predefined minimal support; and,
- Use the frequent items to generate the desired rules.

Specifically, let $I = i_1, i_2, i_3, , i_n$ be a set of items. Let $T = t_1, t_2, t_3, , t_m$ be a set of transactions in the database where $t_i \subseteq I$. The association rules can be simplified in the following form:

$$X \rightarrow Y, where X \subset I, Y \subset I, and \;\; X \cap Y = \emptyset$$

where X and Y are each a set of items called an itemset.

The strength of the rules or the level of interest is measured by support and confidence. In the previous example, the support of a rule, $X \rightarrow Y$, is the number of the transactions I in database T that contains $X \cup Y$. The support of the rule $X \rightarrow Y$ can be generalised as:

$$support = \frac{(|X \cup Y|)}{(|T|)}$$

Support is also defined as the probability $P(X \cup Y)$. Thus, support becomes a useful measure. If the probability of the rule is low, the rule does not occur in many transactions. The largest probabilities represent the rules of most interest to the user.

The other measure that utilises the interest of the rule is confidence. The confidence of rule $X \rightarrow Y$ is the percentage of the transaction in T that $contains(X \cup Y)$. It can also be viewed as the conditional probability, $Pr(X|Y)$. Confidence

TABLE 2.1: Transaction database.

t_1	Beef, Chicken, Milk
t_2	Beef, Cheese
t_3	Cheese, Boots
t_4	Beef, Chicken, Cheese
t_5	Beef, Chicken, Clothes, Cheese, Milk
t_6	Chicken, Clothes, Milk
t_7	Chicken, Milk, Clothes

determines the predictability of the rule, which is computed as follows:

$$confidence = \frac{(|X \cup Y|)}{(|X|)}$$

The main objective of the association rule approach is to discover all of the association rules in given transaction datasets that have support and confidence greater or equal to that of the user's predefined minimum support and minimum confidence ($minsup$ and $minconf$, respectively).

Example: Table 2.1 shows a simple breakdown of seven transactions in shopping baskets. Each transaction represents a set of items purchased in a store by a customer. Assume that the user-specified $minsup = 30\%$ and the $minconf = 80\%$. The following rules are valid, as its support is $42.84\% > 30\%$ and its confidence is 100% 80%.

$$Chicken, Clothes \rightarrow Milk \qquad [sup = \frac{3}{7}, conf = \frac{3}{3}]$$

Text documents can also be treated as transactions at different levels (e.g. sentence, paragraph, or document levels) without considering the word order or sequence. Applying the association rule method to a set of documents or sentences facilitates the identification of frequently co-occurring words.

Association rules have different algorithms, each of which requires user-specified minimum support and confidence. Any algorithm should find the same set of rules, although the computational method may vary. One very traditional algorithm is the Apriori algorithm. The Apriori algorithm consists of two steps:

- Generate all frequent itemsets that have transaction support above the minimum support level; and
- Generate all confidence association rules from the frequent itemsets, as this will reveal the presence of confidence values above the minimum confidence level.

2.2.2 Sequential Patterns

Association rules mining does not consider the sequence of the items that co-occur frequently in a dataset. In many applications, considering the transaction order is significant. For example, in market basket analysis; it is of interest to know whether people buy some items in sequence (such as, buying a bed then buying bed sheets some time later). In text mining, considering the sequence of the words in a transaction is vital for finding language patterns. However, association rule mining will not be appropriate, because it does not consider the order of the transactions.

Sequential pattern mining (SPM) is a data mining method, with broad applications, which attempts to find the relationships among occurrences of sequential items while maintaining their order to extract frequent sequences. The mining results of sequential pattern mining of transactions in text documents are lists of words that appear together in a certain order frequently in the same paragraph [108].

The problem of mining sequential patterns and its main concepts can be described as follows: Let $T = t_1, t_2, , t_k$ be a set of all terms. A sequence $S = \langle s_1, s_2, \cdots, s_n \rangle (s_i \in T)$ is an order of list of terms including duplicate terms. a sequence, $\alpha = \langle a_1, a_2, , a_n \rangle$ is a sub-sequence of another sequence $\beta = \langle b_1, b_2, , b_m \rangle$, denoted by $\alpha \subseteq \beta$, If integers exist $1 \leq i_1 < i_2 < \cdots < i_n \leq m$, such that $a_1 = b_{i1}, a_2 = b_{i2}, , a_n = b_(i_n)$. In SPM the order of terms plays a significant role in deciding when to extract a sub-sequence from a sequence. To clarify, sequence $\langle s_1, s_2 \rangle$ is a sub-sequence of sequence $\langle s_1, s_2, s_3 \rangle$, however, $\langle s_2, s_1 \rangle$ is not. It can be said that sequence $\langle s_1, s_2, s_3 \rangle$ is a super-sequence of $\langle s_1, s_2 \rangle$. The PTM mining sequential patterns was used to find a complete set of sub-sequences from a set of sequences whose support exceeded a predefined minimal support threshold.

In order to solve this problem, a variety of algorithms have been proposed such as PrefIxSpan [48], AprioriAll [61], CloSpan [97], FP-tree [23], SPADE [103], SLPMiner [76], TSP [86], SPAM [6], GSP [48], GST [26], MILE [13] and Sliding Window [46]. Each algorithm pursues a different method of discovering frequent sequential patterns. This makes the algorithm featured simply by the capability of mining such patterns without even generating any candidates (e.g. prefixSpan).

TABLE 2.2: Association rule mining algorithms [89].

Method	Pattern	Algorithm
ApnonAll	sequential	Apriori-like
PrefixSpan	sequential	Apriori-like
FP-tree	sequential	FP-tree
SPADE	sequential	Apriori-like
SLPMiner	sequential	Apriori-like
TSP	closed sequential	Apriori-like
CloSpan	closed sequential	Apriori-like
SPAM	sequential	Apriori-like
GSP	sequential	Apriori-like
GST	sequential	Graph
MILE	sequential	Apriori-like
CLOSET	closed itemset	Apriori-ljke
CLOSE	closed itemset	Apriori-like
CHARM	closed itemset	Apriori-like
GenMax	closed itemset	Apriori-like

2.2.3 Frequent Itemsets

The main difference between a frequent itemset and a sequential pattern is that the order of the items is a concern in the sequential pattern. However, most of the sequential pattern algorithms use data mining association rules to discover frequent itemsets. Apiori is an algorithm that is widely used to extract frequent itemsets [3]. Different algorithms have been used to discover different patterns, as shown in the Table 2.3.

2.2.4 Knowledge Discovery

There is some confusion about the terms "data mining" and "knowledge discovery". Many researchers use "data mining" as a synonym for "knowledge discovery", but data mining is also one step in the knowledge discovery process

(KDP). Moreover, data mining is also known under many other names, such as "knowledge extraction" and "data pattern processing".

The large amount of data being collected in databases today exceeds our ability to analyse and extract the most useful information without the aid of automated analysis techniques. Knowledge discovery is the process of non-trivial extraction of implicit, unknown, and potentially useful information from a large database. Data mining has been used in KD to discover patterns that correspond to a user's needs. "Pattern definition" is an expression that describes a subset of data. An example of a KD pattern definition appears in [89].

$$\textbf{IF}(Age > 30 \ \ \textbf{and} \ \ Salary > 70000)$$
$$\textbf{Then}\ buy(BMW)$$
$$With\ confidence\ (\textbf{0.6 ... 0.8})$$

The above pattern can be simply understood and used directly by a knowledge discovery system (e.g. expert system). The accurate discovery of patterns through data mining is shaped by several factors, such as the size of a sample, the integrity of the data, and the support available from domain knowledge. These factors all affect the degree of certainty of the identified patterns. Typically, data mining uncovers a number of patterns in a database, but only some of them are of interest to the user. Useful knowledge is comprised of the patterns that are of interest to the user. It is important for users to consider the degree of confidence in a given pattern when evaluating its validity. The validity of the discovered pattern can be defined by either the system or the user.

In summary, knowledge discovery must exhibit the following characteristics:

- **Interestingness**: Discovered knowledge is deemed interesting based on the implication that patterns should be novel and potentially useful, and the process of knowledge discovery must be nontrivial.
- **Accuracy**: Discovered patterns should accurately depict the contents of the data. The extent to which this depiction is imperfect is expressed by measures of certainty.
- **Efficiency**: The process of knowledge discovery is efficient, especially for large data sources. An algorithm is considered efficient if the run time is acceptable and predictable.
- **Understandability**: High-level language is required to express discovered knowledge and the expression must be understandable to users.

2.2.4.1 Knowledge Discovery Process

The knowledge discovery process (KDP) is defined as the nontrivial process of extracting understandable, useful, novel, and valid patterns from row data. The KDP consists of many steps executed in sequence, with data mining being one of those steps. Loops can occur between any two steps in the process for further iteration. Inputs include data in different formats, such as numerical or nominal data stored in databases or flat files. The subsequent step uses the previous steps output result as an input. Knowledge discovery concerns the entire knowledge extraction process, from storing and accessing data to visualising results and supporting interactions between machine and user. The final output is usually described in patterns, association rules, statistical analyses, classification models, etc. A schematic diagram is shown in the following Figure 2.2.

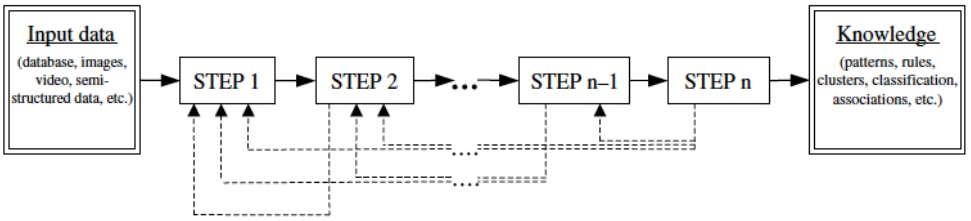

FIGURE 2.2: Sequential structure of the KDP model [16].

Since the 1990s, several different KDPs have been developed by academic researchers and quickly adopted by the industry. According to Krzysztof J. Cois and other researchers, the first basic structure of the model was developed by Fayyad et al. in the mid-nineties. The two process models he developed and proposed in 1996 and 1998 comprise the nine-step model now perceived as the leading research model. The nine steps of the Fayyad et al. model can be outlined as follows [16]:

- The first step in the KD process is to develop an understanding of the application domain, including relevant prior knowledge, and identify the goal of the end user.
- Second, one chooses a target data set by selecting a data set and focusing on the subset of variables (attributes) or data samples that will be used to perform discovery tasks.
- Third, the data is cleaned and pre-processed by reducing noise, designing strategies for dealing with missing data, and accounting for time-sequence information and known changes.
- The fourth step encompasses the data reduction and projection phase in which one finds useful features to represent the data, including the possible

use of dimensionality reduction or transformation methods to present the data.

- In the fifth step, the data miner matches the goals defined in step 1 with a particular data mining method, such as classification, clustering, ... etc.
- Sixth, choosing the right algorithm for a specific task can be a challenge, as many different algorithms can be used to perform the same task. In this step, the dataset is matched with data mining algorithms to search for patterns.
- The seventh step is the data mining step, which generates patterns in a particular representational form, such as classification rule, decision tree, etc.
- In the eighth step, the analyst visualises the extracted patterns and models, and represents the data based on the extracted model.
- For the ninth and final step, the user incorporates the discovered knowledge into the performance system, and documents or reports it to the interested parties.

The KD process is interactive. It examines many decisions made by the user, and can involve significant iteration and contain loops between any two steps. However, most of the previous work has focused on step 7, which is data mining. The basic flow of steps (although not the potential multitude of iterations and loops) is illustrated in Figure 2.3.

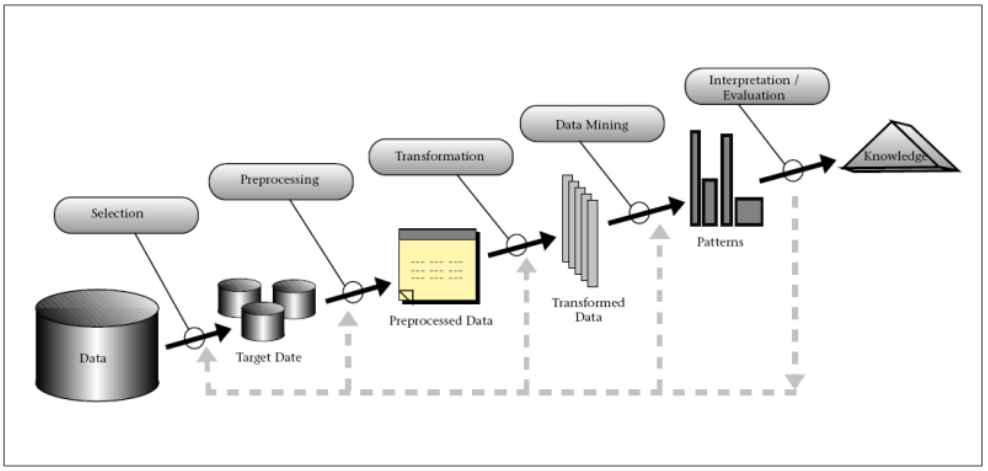

FIGURE 2.3: Steps in the KD process [20].

2.2.5 Web Mining

Web mining was developed from data mining methodologies. Data mining is the process of extracting useful information and patterns from large amounts of data. Web mining utilizes data mining algorithms specifically to extract information from Web documents and services [40]. It takes into account the complexity of Web structures, where the data available in "www" form is heterogeneous, semi-structured, and un-structured, as well as the diversity of Web documents. Thus, not all data mining algorithms can be used in Web mining. This is a vast area of research that continues to garner more and more attention. Web mining taxonomy is categorised into three main areas: Web content mining, Web structure mining, and Web usage mining.

Web content mining is the process of extracting useful information and patterns from the content of Web sources, including text and multimedia data, such as HTML files, images, or email. Web content mining is related to, but differs from, data mining and text mining. It is related to data mining because most

data mining techniques can be applied to Web content and it differs from data mining because Web data are mainly semi-structured or unstructured, while data mining applies primarily to structured data. It is related to text mining because Web content is comprised of so much text. However, it is also quite different from text mining due to the semi-structured nature of the Web. Text mining focuses on unstructured text and is typically used for summarising, categorising, clustering, and association analysis, as well as all analysis that extends beyond keyword extraction or utilises some statistical word or phrase methods [37]. Web content mining thus requires creative applications of data mining and text mining techniques, as well as its own unique approaches.

Web structure mining uses the structure of the Web as an additional source of information. This category collects both the uncontrolled heterogeneous documents and the hyperlinks. Because Web pages connect to other pages containing related content, the hyperlinks contain a large amount of latent human annotations that can be useful in different ways [40]. Web structure mining can also be helpful in discovering the structure of a Web document. Understanding a Web page schema enables a researcher to use database techniques to access information embedded in Web pages [50]. The main objective of Web structure mining is to generate a structural summary of the Web site and Web page. Based on the topology of the hyperlinks, Web structure mining classifies Web pages and generates information, such as the relationship between different Web sites. Discovering the structure of a Web document can help to reveal the schema of Web pages, which is useful for navigation and Web page scheme integration.

Web usage mining examines user access patterns, such as which pages are frequently accessed. Web usage mining data is comprised of the Web user's browsing

history, browser logo, cookies, and other sources. Extracting user profile data via Web usage mining can facilitate Web personalisation and site structure modification [40]. The objectives are to extract, model, and analyse the behavioural patterns and profiles of users interacting with a Web site. The discovered patterns are generally represented as collections of pages, objects, or resources that are frequently accessed by groups of users with common needs or interests. Following the standard data mining process, the overall Web usage mining process can be divided into three interdependent stages: data collection and pre-processing, pattern discovery; and pattern analysis [48].

2.3 Information Retrieval

Information retrieval (IR) is defined as the process of finding useful information in documents, databases, and other sources. Searching for information on the Web is a complicated task because Web documents are semi-structured or non-structured.

Since 1960, extensive work has been conducted on topics such as indexing. Researchers successfully applied IR methodologies to the Web, which gave rise to search engines. However, Web search engines face unique challenges that databases do not. First, due to the vast amount of documents available on the Web, search engines are only able to index a fraction of the available information. Second, search engines return a large number of documents that include user keywords, but not all of these documents are relevant to the search topic. Given the rapidly increasing amount of information sources available on the Web, the IR

community faces the challenge of managing a huge amount of hyperlinked data, but members of this community can utilise modelling, document classification and categorisation, user interfaces, and data visualisation filtering to accomplish their goals [8].

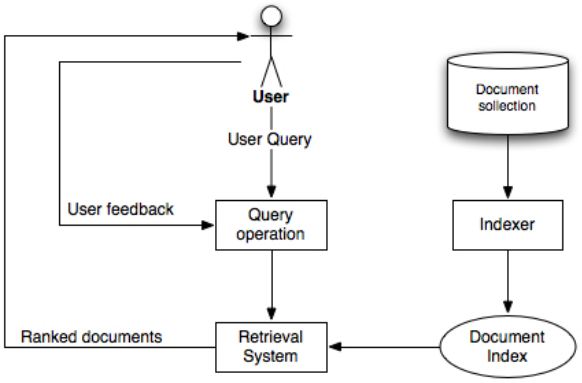

FIGURE 2.4: A general IR system architecture.

As shown in Figure 2.4, the user needs to submit a query to the retrieval system through the query operations model. The retrieval model uses the document index to retrieve the documents that are likely to be relevant to the user, based on the keyword-matching approach. The scores of each retrieved document will be calculated and the documents will be ranked according to these scores. Finally, the ranked documents are presented to the end user [48].

2.3.1 Feature Selection

One of the objectives of knowledge extraction is to build user profiles by finding a set of features from feedback documents to describe what a user needs. This is a particularly challenging task in modern information analysis, from both empirical

and theoretical perspectives [44, 47]. This problem has charged much attention from researchers in Data Mining, Web Intelligence and Information Retrieval (IR) communities.

There will be a large number of terms extracted from text using data mining methods. The high dimensionality of the feature space leads to computational complexity and overfitting problems. The simple way to reduce the dimensionality is the filtering approach, which filters irrelevant terms based on the measures derived from the statistical information. Feature selection has been an active research area in pattern recognition, statistics, and data mining communities. Feature selection can significantly improve the comprehensibility of the results by reducing the dimensionality.

Over the years, IR has developed many mature techniques that demonstrate the usefulness of term-based features in text documents. However, many feature terms with larger weights appear to be general terms because they are frequently used in both relevant and irrelevant samples. For example, the term "LIB" may have larger weight than "JDK" in a data collection; but we believe that the term "JDK" is more specific than the term "LIB" for describing "Java Programming Language"; and the term "LIB" is more general than the term "JDK" because term "LIB" is also frequently used in C and C++.

Many types of text representation have previously been proposed. A well-known example is the bag-of-words that uses keywords (terms) as elements in the vector of the feature space. In [38], the TF-IDF weighting scheme was used for text representation in Rocchio classifiers. Enhanced from TF-IDF, the global IDF and entropy weighting scheme proposed by Dumais [18] improved performance by an

average of 30%. Various weighting schemes for the bag-of-words representation were given in [1, 29, 71]. The problem of the bag-of-words approaches is how to select a limited number of feature terms in order to increase the system's efficiency as well as avoid *overfitting* [75]. In order to reduce the number of features, many dimensionality reduction approaches have been conducted using feature selection techniques, such as Information Gain, Mutual Information, Chi-Square, and Odds ratio [75].

2.3.1.1 Term Frequency

The frequency of a term t in a document d can be used for document-specific weighting and denoted as $TF(d, t)$. It is only a measure of a term's significance within a document.

2.3.1.2 Inverse Document Frequency

Inverse Document Frequency (IDF) is used to measure the specificity of terms in a set of documents. It assumes that a high-semantical term appears in only a few documents, while a low-semantical term is spread over many documents. The formula of IDF can be expressed by the following:

$$IDF(t) = log\ \frac{|D|}{DF(t)}$$

where D is the set of documents in the collection and $DF(t)$ is the document frequency, which is the number of documents where the term t appears at least once.

2.3.1.3 Term Frequency Inverse Document Frequency

Term Frequency Inverse Document Frequency (TF-IDF) [71] is the most widely adopted measure. TF-IDF is the combination of the exhaustivity statistic (TF) and the specificity statistic (DF) to a term.

$$TFIDF(t) = TF(d,t) \times IDF(t)$$

2.3.1.4 Residual Inverse Document Frequency

Residual Inverse Document Frequency (RIDF) is a variation of IDF. RIDF assigns collection-specific measures to terms according to the difference between the logs of the actual IDF and the prediction by a Poisson model [15]. It measures the distributional behaviour of terms across documents. The function of RIDF is expressed in the following:

$$RIDF(t) = IDF(t) + log(1 - Prob_p(t))$$

where $Probp(t) = 1p(0;\lambda)$ is the Poisson probability that t appears at least once in a document; $\lambda = CF(t)/N$ is the average number of occurrences of terms t per document.

2.3.1.5 Relative Frequency Technique

Relative Frequency Technique (**RFT**) is suggested in [55] with the assumption that special or technical words are more rare in general usage than in documents

about the corresponding subjects. In contrast to pure **TF**, **RTF** uses a terms collection statistic.

$$RFT(t) = \frac{TF(d,t)}{T_D}\frac{CF(t)}{T_c}$$

where $CF(t)$ is the collection frequency denoting the number of times a term t appears in the entire collection. T_d and T_c are the total number of terms in the document and the number of terms in a general document collection respectively.

2.3.2 Term Weighting

Term weighting is one of the most important components of the information filtering and retrieval system. Term weighting estimates significance weights for terms, based on statistical information. The weight of a term indicates how the term applies to a topic by exploiting the statistical variations in the distribution of terms within the relevant documents and within the complete document collection [55]. Many of the proposed weighting methods include TF-IDF, which has previously been described. In the following section, we discuss some popular weighting methods.

2.3.2.1 Probabilistic Model

Stephen and Karen Sparck [81] proposed four probabilistic functions for term weighting, based on the binary independence retrieval model. Two types of assumptions are used in these functions: independence assumptions and ordering

principles. The four probabilistic functions can be described as follows:

$$F_1(t) = log\ \frac{(\frac{r}{R})}{(\frac{n}{N})}$$

$$F_2(t) = log\ \frac{(\frac{r}{R})}{(\frac{n-r}{N-R})}$$

$$F_3(t) = log\ \frac{(\frac{r}{R-r})}{(\frac{n}{N-n})}$$

$$F_4(t) = log\ \frac{(\frac{r}{R-r})}{(\frac{n-r}{N-n-R+r})}$$

where r is the number of relevant documents that contain term t, n is the total number of documents in the collection that contain term t, R is the total number of relevant documents, and N is the number of documents in the collection.

2.3.2.2 Okapi Model (BM25)

The Okapi model is based on the aforementioned probabilistic model. The BM25 function in the Okapi model utilises term frequency and document length [30, 67]. The weighting function BM25 can be generalised as:

$$BM25 = \frac{TF(k_1+1)}{k_1 NF + TF}\ log\ \frac{(r+0.5)(N-n-R+r+0.5)}{(R-r+0.5)(n-r+0.5)}$$

$$NF = (1-b) + b\ \frac{DL}{AVDL}$$

where TF is the term frequency, k_1 and b are the tuning parameters, and DL and $AVDL$ denote the length of the document and the average document length, respectively.

2.3.3 Information Retrieval Models

An IR model governs how to present the query and relevant documents and how to define these documents to a user. There are four main IR models: the Boolean model, the vector space model, the statistical language model, and the probabilistic model. The first three models are the most commonly used models, and all three treat each document and query as a list of words or terms, ignoring the sequence of the words or terms. A list of terms in a document d can be donated as: $I = t_1, t_2, t_3, , t_n$, where t_n is usually a word. Thus, each document d_j is represented with term vectors as [48]:

$$d_j = (w_1 j, w_2 j, w_3 j, , w_i j)$$

where $w_i j$ is the weight of term $t_i \in d_j$.

Each term is an attribute and each weight is an attribute value. The method of calculating attribute values in the IR model differs from that of the other models.

2.3.3.1 Boolean Model

The Boolean model is a very simple and old information retrieval model. This model is used to calculate the weight of the terms and documents. Calculating the weight of the documents helps to match the documents with the user queries. Both the documents and the queries are represented as vectors based on a set of terms or words. The weight of each term t_i is calculated with the following

equation:

$$w_{ij} = \begin{cases} 1 \; if \; t_i \; \in d_j \\ \\ 0 \; otherwise \end{cases}$$

Boolean queries combine query terms logically using Boolean operators, such as AND, OR, and NOT. For example, the query ((*x* AND *y*) AND (NOT *z*)) means the retrieval documents must contain the words x and y, but not z. The system's retrieval of the document makes the query logically true, based on the binary decision.

2.3.3.2 Vector Space Model

According to Bing, this model is probably the most widely used information retrieval model. In this model, the document is represented as a weight vector. Each attribute weight is computed based on some variation of the TF or TF-IDF scheme. Thus, the weight of each term can be any number.

Term frequency schema (TF): the weight of the term t_ij in document d_j is the number of times that t_ij occurred in document d_j:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$$

where n is the number of times the considered term occurs in document d_j.

TF-IDF: TF is still the term frequency in the document and IDF is the inverse document frequency. This scheme is well-known and widely used:

$$idf_i = log \frac{|D|}{|d_j : t_(i)d_j|}$$

$$TF - IDF = tf_{ij} * idf_i$$

where $|D|$ is the total number of documents in the corpus and $|d_j : t_(i)d_j|$ is the number of documents containing the term t_i, as illustrated in [48].

2.3.3.3 Statistical Language Model

Statistical models, also called "language models", are based on probability. The idea behind the statistical model approach can be described through the following steps. First, estimate the language model for each document. Second, rank each document based on the likelihood of the query.

Let the query q be a sequence of terms $q = q_1, q_2, , q_m$, and the document collection D be a set of documents, $D = d_1, d_2, , d_n$. In the statistical language model, the probability of a query q being generated by a probabilistic model is based on a document $d_j, Pr(q|d_j)$. When ranking documents in retrieval, we are interested in estimating the future probability $Pr(q|d_j)$. The available Bayes rule is used for ranking [48].

$$Pr(d_j q) = \frac{Pr(q|d_j)Pr(d_j)}{Pr(q)}$$

2.3.4 Information Filtering

Information filtering (IF) and information retrieval (IR) work side by side. However, information filtering is used to decrease the amount of irrelevant data from a stream of incoming data. Or in other words, the objective of information filtering is to reduce the users' information load with respect to their areas of interest. Information filtering can be viewed as a specific form of supervised text classification and is used to make a decision (yes or no) for every document as soon as it arrives, with respect to the pre-defined topic of interest. Generally speaking, a filtering system computes a score for every document based on the user's feedback and then establishes the thresholds the scores must meet to help determine whether the document should be included in the final list of results or disregarded [102].

Several approaches apply data mining techniques to extract knowledge from Web logs in order to facilitate personalisation tasks. However, some researchers have pointed out that Web usage data from the server side are not always reliable [82]. Nonetheless, some information filtering systems use explicit user feedback, such as the two-stage model, which, in the first stage, aims to quickly filter irrelevant information based on the user profiles. In the second stage, the researcher applies data mining techniques to rationalise the relevance of the reduced data set [110]. As far as we know, there are three types of Web search systems that provide such information with respect to user need: systems using relevance feedback, systems where users register their interests or demographic information, and systems that recommend information based on user ratings, where users have to

register personal information, such as their interests, age, etc., beforehand, or provide positive or negative feedback [82].

Traditional information filtering models were developed around a term-based user profile approach (see [19, 54, 63, 109]), most of these models extracted terms from the positive training documents only. It was assumed that the phrase- (pattern-) based approaches would perform better than the term-based approaches, as patterns carry more semantic information than terms [45]. However, many experiments have found the opposite to be true [90]. The most likely reason that has been given is that using a long pattern is less useful, we argue, however, that the longer the pattern, the lower its frequency of occurrence. Statistically, patterns have inferior properties to words [45]. Therefore, in some cases, using terms can be more useful than using patterns.

The pattern taxonomy model (PTM) is one of the new models based on pattern methodology. The PTM works in two stages. The first stage extracts useful phrases from text documents. The weight of the terms occurring in the extracted patterns is then calculated to improve judgements on new documents [25]. Currently, the PTM system uses positive feedback and ignores negative feedback, as most information filtering systems do.

2.3.5 Adaptive Information Filtering

An information filtering (IF) system monitors the incoming document stream and makes a binary decision to accept or reject documents based on user profiles. The IF system can be viewed as an interactive learning process because the user is able to provide feedback to the system. In contrast, it is a non-interactive

TABLE 2.3: Information filtering models [89].

IF Model	Representation	Term weighting
KerMIT	bag-of-words	Kernel Function
PIRCS	bag-of-words	TFIDF
Okapi	bag-of-words	TFIDF
RhLIEFS	bag-of-words	Probabilistic
Rutgers	bag-of-words	TFIDF
CLARIT	bag-of-words	TFIDF
NewT	bag-of-words	TFIDF
NewsWeeder	bag-of-words	TFIDF
Aplipcs	bag-of-words	TFIDF
GroupLens	bag-of-words	TFIDF
INFOrmer	nGram	TFIDF
SIFT	bag-of-words	TF
ProFile	bag-of-words	TFIDF
INQUERY	bag-of-words	Probabilistic

machine learning problem with a set of labelled documents provided in advance. The task of IF can be regarded as a special instance of text classification. The historical development of IF can be seen in [87]. Unlike a traditional information retrieval system, an adaptive filtering system maintains user profiles that tend to reflect a need for long-term information. By interacting with users, an adaptive filtering system can build a better profile and update it based on feedback to improve its performance over time. The main idea of the adaptive system is for users to obtain relevant documents as soon as they arrive and filter out all irrelevant documents. Lau et al. [33] applied belief revision logic to model the task of adaptive information retrieval. Landquillon [12, 31] proposed two methods for assessing performance indicators in the absence of user feedback. Table 2.3 shows the existing information filtering systems from the related literature. Most of these systems adopted singular words (known as "bags of words") for data representation and the $TF - IDF$ variant for the term weighting scheme.

An adaptive information filtering system can be studied from two points of view. First, is the ability of IF system to extract different knowledge for different users in different interested topics. Since, each user has different information needs in regard to the provided query. For example, for the query "Apple", some users may be interested in documents dealing with "Apple Inc", while other users may want documents related to "Apple fruit". Relevance feedback can be used to adapt Web search results for users with different information needs. Second, is the ability to update and review the weight of features in the hypothesis space model when new feedback is received. Traditionally, information involved in knowledge-based systems has been manually acquired in collaboration with domain experts. However, both the high cost of such a process and the existence of textual sources containing the required information have led to the use of automatic acquisition approaches. In the early eighties, text based intelligent (TBI) systems began to manipulate text so as to automatically obtain relevant information in a fast, effective, and helpful manner [85]. In reality, the system starts with a limited number of feedback responses though it will receive more later on. As a result, the extracted information needs to be updated. There are two ways to update the learning model with new feedback documents:

- Via batch updates, when new feedback is received, the user's profile is rebuilt using a combination of the old and new feedback datasets. This kind of update is time-consuming and needs more space to store all feedback received.

- Adaptively, by extracting information from the new feedback documents and merging it with existing information stored in the system. This method

is very efficient but it is more complicated than the previous one.

The main aim of Adaptive Information Filtering system is to automatically fillers incoming data streams to topics a specific user is interested in [84]. The filtering process is done based on some knowledge that system has extracted from feedback documents. Providing more feedback documents can lead to increased the amount of knowledge that system has about what user need. The new feedback provided can be grouped into two main categories:

- The first involves updates of the user's profile to follow changes in the user's interests. For example, if the user started searching about "Java programming language" and then moved to a different area of search like searching for information about "Australia". This issue is out scope of this research.

- The second area involves updating the user profile to solve nonmonotonic problems. For instance, given the system's limited number of training documents about the term "Agent", the IF system may return information objects such as "Intelligent Agent", "Property Agent", or "Software Agent". However, when more information about the user's actual information needs is obtained later on, the system can determine that the user is only interested in "Software Agent" [32].

Many algorithms have been developed to deal with this issue. These algorithms are mainly focusing on knowledge extracted automatically, set system parameters, calibration and optimisation of thresholds or on updating the users' profiles to follow changes in interest [49, 82, 85, 100, 101]. These problems have been

studied in the adaptive filtering area and include topic profile adaptation using incremental Rocchio, Gaussian-Exponential density models, logistic regression in a Bayesian framework, etc., and threshold optimisation strategies using probabilistic calibration or local fitting techniques [10, 73, 100, 101, 105, 106]. However, it seems that threshold optimisation problems are no longer an issue. This is because the aim of IF is to re-rank the incoming set of documents based on the user profile rather than making Yes or No decisions for each document.

Using a ranking algorithm to rank a list of documents based on knowledge extracted from relevance feedback documents shifts the focus of adaptive approaches from optimising the threshold, to how to use the new feedback documents to update knowledge of the system. The main focus in this research is how to update the system with this new knowledge effectively and efficiently. Some issues need to be considered when updating the system with new knowledge. These issues can be generalised as:

- does the new relevance feedback contain new knowledge?
- how to select the documents that contain new knowledge?
- how to merge the new knowledge with the existing knowledge held by the system?
- how to evaluate the merging process to ensure the new knowledge does not conflict with the existing knowledge?

2.4 Text Mining

Text mining, sometimes called "text data mining", is the process of text categorisation, text clustering, concept extraction, granular taxonomy production, sentiment analysis, document summarisation, and entity relation modelling in text data. A large portion of available data appears in the collection of text data, be it structured, semi-structured, unstructured, or even mixed data. Text databases usually consist of collections of articles, research papers, emails, blogs, discussion forums, and Web pages. A very large number of features (e.g. terms) can be extracted from the text database to describe each text or document. The high dimensionality of the feature space leads to over fitting problems. To reduce the amount of extracted terms, only valuable terms are selected.

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 [75]. To date, many classification methods, such as Naive Bayes, Rocchio, kNN and SVM have been developed in IR [51].

SVM is one of the main machine learning methods for text classification [38, 95]. It also performed better on Reuters data collections than kNN and Rocchio [36, 99]. The classification problems include the single labelled and multi-labelled problem. The most common solution [111] to the multiple labelled problem is to decompose it into independent 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 [4, 27, 35]. However, the effectiveness of the text mining systems using phrases to represent text showed no significant improvement due to the large amount of incoming training data for an individual category (or a query), therefore phrases are of less help. Recently a concept-based model that analyses terms on the sentence and document levels was introduced in [77], and ontology mining methods [44, 83] also provided some thoughts regarding text classification. However, they usually relied upon employing 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 [24, 26, 76, 97, 103] 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 duplicate terms, and labelled information in the training set [44]. 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 [93]. Nevertheless, the challenging issue is how to effectively deal with the large amount of discovered patterns and terms which contain a lot of noise.

2.4.1 Pattern Taxonomy Model (PTM)

A promising techniques were pattern taxonomy model (PTM) [28, 90, 91] that discovered closed sequential patterns in text documents, where a pattern was a set of terms that frequently appeared in paragraph. These approaches introduced data mining techniques to information filtering; however, too many noisy patterns adversely affect the PTM systems [45]. The PTM, like many filtering systems, is more reliable using positive training documents only.

In PTM, all documents are split into paragraphs. So a given document d yields a set of paragraphs $PS(d)$. Let D be a set of feedback 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^+.

2.4.1.1 Sequential Pattern Mining (SPM)

The basic definitions of sequences used in this research work are described as follows. Let $T = \{t_1, t_2, ..., t_k\}$ be a set of all terms, which can be viewed as words or keywords in text documents. A sequence $S = \langle s_1, s_2, ..., s_n \rangle (s_i \in T)$ is an ordered list of terms. Note that the duplication of terms is allowed in a sequence. This is different from the usual definition where a pattern consists of distinct terms.

A sequence $\alpha = \langle a_1, a_2, ..., a_n \rangle$ is called sub-sequence of another sequence $\beta = \langle b_1, b_2, ..., b_m \rangle$, denoted by $\alpha \sqsubseteq \beta$, if there exist integers $1 \leq i_1 \leq i_2 < ... < i_n \leq m$, such that $a_1 = b_{i_1}, a_2 = b_{i_2}, ..., a_n = b_{i_n}$.

For instance, sequence $\langle s_1, s_3 \rangle$ is a sub-sequence of sequences $\langle s_1, s_2, s_3 \rangle$. However, $\langle s_2, s_1 \rangle$ is not a sub-sequence of $\langle s_1, s_2, s_3 \rangle$ since the order of terms is considered. The sequence α is a proper sub-sequence of β if $\alpha \sqsubseteq \beta$ but $\alpha \neq \beta$, denoted as $\alpha < \beta$. In addition, we can also say sequence $\langle s_1, s_2, s_3 \rangle$ is a super-sequence of $\langle s_1, s_3 \rangle$. The problem of mining sequential patterns is to find a complete set of sub-sequences from a set of sequences whose support is greater than a user-predefined threshold, *minmum_sup*. Pattern taxonomy is a tree-like hierarchy that reserves the sub-sequence (i.e."is-a") relationship between discovered sequential patterns.

Two kinds of supports are used to calculate the support of patterns. Given a document $d = \{S_1, S_2, ..., S_n\}$, where S_i is a sequence representing a paragraph in d. Thus, $|d|$ is the number of paragraphs in document d. Let P be a sequence. Pattern P is called a sequential pattern of d if there is a $S_i \in d$ such that $P \sqsubseteq S_i$. The absolute support of P is the number of occurrences of P in d, denoted as:

$$supp_a(P) = |\{S | S \in d, P \sqsubseteq S\}|$$

The relative support of P is the fraction of paragraphs that contain P in document d, denoted as:

$$supp_r(P) = supp_a(P)/|d|$$

For example, the sequential pattern $P = \langle t_1, t_2, t_3 \rangle$ in the sample database, as shown in Table 2.4, has $supp_a(P) = 2$ and $supp_r(P) = 0.5$. All sequential patterns in Table 2.4 with absolute support greater than or equal to 2 are presented in Table 2.5.

TABLE 2.4: Each transaction represents a paragraph in a text document and contains a sequence consisting of an ordered list of words.

Transaction	*Sequence*
1	$S_1 : \langle t_1, t_2, t_3, t_4 \rangle$
2	$S_2 : \langle t_2, t_4, t_5, t_3 \rangle$
3	$S_3 : \langle t_3, t_6, t_1 \rangle$
4	$S_4 : \langle t_5, t_1, t_2, t_7, t_3 \rangle$

TABLE 2.5: All frequent sequential patterns discovered from the sample document (Table 2.4) with *minmum_sup*: $\xi = 0.5$.

Patterns	$supp_a$	$supp_r$
$\langle t_4 \rangle, \langle t_5 \rangle, \langle t_1, t_3 \rangle, \langle t_2, t_4 \rangle, \langle t_5, t_3 \rangle, \langle t_1, t_2, t_3 \rangle$	2	0.5
$\langle t_1 \rangle, \langle t_2 \rangle, \langle t_2, t_3 \rangle$	3	0.75
$\langle t_3 \rangle$	4	1

The relative support of a pattern is used to properly estimate the significance of the pattern in the document. Usually, a pattern with the same frequency will acquire the same support in different document lengths. However, with the same occurrence frequency, a pattern is more significant in a short document than in a long one. In contrast to other approaches, we decompose a document into a set of transactions and discover frequent patterns from these by using data mining methods. The relative support of a pattern can be estimated by dividing absolute support by the number of transactions in a document. Hence, a pattern can obtain an adequate support for various document lengths with the same frequency.

A sequential pattern P is called a frequent sequential pattern if $supp_r(P)$ is greater than or equal to a minimum support (*minmum_sup*) ξ.

For example, let *minmum_sup* be 0.75 for mining frequent sequential patterns

from a sample document in Table 2.4, Then we can obtain four frequent sequential patterns: $\langle t_2, t_3 \rangle$, $\langle t_1 \rangle$, $\langle t_2 \rangle$, and $\langle t_3 \rangle$ since their relative supports are not less than ξ as shown in Table 2.5.

The length of sequential pattern P, denoted as $len(P)$, indicates the number of words (or terms) contained in P. A sequential pattern which contains n terms can be denoted in short as a $nTerms$ pattern. For instance, given pattern $P = \langle t_2, t_3 \rangle$, we have $len(P) = 2$, and P is a $2Terms$ pattern. A sequential pattern may consist of several terms (words) or one term only. Thus, a $1Term$ pattern is a special sort of $nTerms$.

2.4.1.2 Closed Sequential Patterns

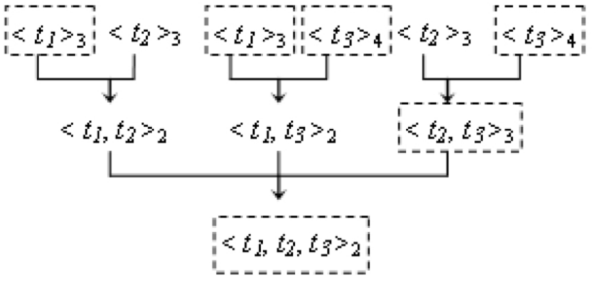

FIGURE 2.5: An example of pattern taxonomy where patterns in dash boxes are closed patterns.

Normally not all frequent patterns are useful [91, 93]. The PTM defines closed patterns as meaningful patterns as most of the sub-sequence patterns of closed patterns have the same frequency, which means they always occur together in a document. For example, in Figure 2.5, patterns $\langle t_1, t_2 \rangle$ and $\langle t_1, t_3 \rangle$ appear twice in a document as their parent pattern $\langle t_1, t_2, t_3 \rangle$ has a frequency of two. SPM stands for sequential pattern mining and the PTM defines sequential closed

TABLE 2.6: Frequent patterns and covering sets.

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

pattern mining as SCPM. The notion of a closed pattern is defined as follows: a frequent sequential pattern P is a closed sequential pattern if there exists no frequent sequential patterns P' such that $P \sqsubset P'$ and $supp_a(P) = supp_a(P)$. The relation $\sqsubset$ represents the strict part of the subsequence relation $\sqsubseteq$.

For instance, the nodes in Figure 2.5 represent sequential patterns extracted from Table 2.4. Only the patterns within dash-line borders are closed sequential patterns if *minmum_sup*, $\xi = 0.50$. The others are considered as non-closed sequential patterns.

2.4.1.3 Pattern Taxonomy

Patterns can be structured into a taxonomy by using the *is-a* (or *subset*) relation and closed patterns. For example, Table 2.6 contains ten frequent patterns; however, it includes only three closed patterns: $< t_3, t_4, t_6 >$, $< t_1, t_2 >$, and $< t_6 >$. Simply, a pattern taxonomy is described as a set of pattern-absolute support pairs, for example 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 $< t_3, t_6 >$ and $< t_4, t_6 >$.

Smaller patterns in the taxonomy, for example pattern $\{t_6\}$, are usually more general because they have a high occurrence frequency in both positive and negative documents; but larger patterns, for example pattern $\{t_3, t_4, t_6\}$, in the taxonomy are usually more specific since they have a small chance of being found in both positive and negative documents.

2.4.1.4 Deploying Method

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

To improve the effectiveness of the pattern taxonomy mining, an algorithm was proposed in [90] (also used in [45]) to find all closed sequential patterns, which used the well-known *Apriori* property in order to reduce the searching space. For every positive document d, the SPMining algorithm discovered a set of closed sequential patterns based on the $minmum\_sup$.

Let SP_1, SP_2, ..., $SP_{|D^+|}$ denote the sets of discovered closed sequential patterns for all documents in D^+. For a given term, its support in discovered patterns from D^+ can be described as follows:

$$support(t, D^+) = \sum_{i=1}^{|D^+|} \frac{|\{p|p \in SP_i, t \in p\}|}{\sum_{p \in SP_i} |p|} \tag{2.1}$$

Extracting patterns first then deploying them on the term space to calculate term weights would help to reduce the number of noisy terms, and give more accurate weights to terms. The obvious reason is that terms that appear in both short patterns and their super patterns would get larger weights.

Table 2.7 illustrates a real example of pattern taxonomy for a set of positive documents $D^+ = \{d_1, d_2, \cdots, d_5\}$. For example, the 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 pattern taxonomies:

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

TABLE 2.7: Example of a set of positive documents consisting of pattern taxonomies. The number beside each pattern indicates its absolute support, where the minimum absolute support is 2.

Doc.	**Pattern Taxonomies**	**Sequential Patterns**
d_1	$PT_{(1,1)}$	$\{\langle carbon\rangle_4$, $\langle carbon, emiss\rangle_3\}$
	$PT_{(1,2)}$	$\{\langle air, pollut\rangle_2\}$
d_2	$PT_{(2,1)}$	$\{\langle greenhous, global\rangle_3\}$
	$PT_{(2,2)}$	$\{\langle emiss, global\rangle_2\}$
d_3	$PT_{(3,1)}$	$\{\langle greenhous\rangle_2\}$
	$PT_{(3,2)}$	$\{\langle global, emiss\rangle_2\}$
d_4	$PT_{(4,1)}$	$\{\langle carbon\rangle_3\}$
	$PT_{(4,2)}$	$\{\langle air\rangle_3, \langle air, antarct\rangle_2\}$
d_5	$PT_{(5,1)}$	$\{\langle emiss, global, pollut\rangle_2\}$

Algorithm PTM (D^+, *minmum_sup*)
Input: D^+; minimum support, *minmum_sup*.
Output: a set of r-patterns RP, and supports of terms.
Method:
(1) $RP = \emptyset$;
(2) **for** (document $d \in D^+$) {
    let DP be the set of paragraphs in d;
    //sequential pattern mining in a set of paragraphs
    SP = SPMining(DP, *minmum_sup*);
    $\overrightarrow{d} = \emptyset$;
    for (pattern $p_i \in SP$) {
        $p = \{(t, 1) | t \in p_i\}$;
        $\overrightarrow{d} = \overrightarrow{d} \oplus p$ }
    $RP = RP \cup \{ \overrightarrow{d} \}$ }
(3) $T = \{t | (t, w) \in p, p \in RP\}$;
    for (term $t \in T$) $support(t) = 0$;
(4) **for** (r-pattern $p \in RP$)
        for ($(t, w) \in \beta(p)$)
            $support(t) = support(t) + w$;

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 $\tau(t, d) = 0$.

2.4.1.5 Pattern Mining Algorithm

To improve the efficiency of the mining of the pattern taxonomy, an algorithm, *SPMining*, was proposed in [90] to find all closed sequential patterns, which used the well known *Apriori* property in order to reduce the searching space.

The PTM algorithm describes the training process of pattern taxonomy mining. For every positive document, the SPMining algorithm is first called in step (2) giving rise to a set of closed sequential patterns SP. Additionally, all discovered patterns in a positive document are composed into an r-pattern giving rise to a set of r-patterns RP in step (2). Thereafter in step (3) and (4), the support is calculated for all terms that appear in the r-patterns, where the normal forms $\beta(p)$ for all r-patterns $p \in RP$ is used.

After the support of terms have been computed from the training set, a given pattern's support to the given topic can be defined as follows:

$$support(p) = \sum_{t \in p} support(t).$$

It is also easy to verify $support(p_1) \leq support(p_2)$ if p_1 is a sub-pattern of pattern p_2. This property shows that a document should be assigned a large weight if it contains large patterns. Based on this observation, we will assign the following weight to a document d for ranking documents in the second stage:

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

2.4.2 Relevance Feedback

Information Retrieval (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 [98], 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.

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, the term "LIB" may have larger weight than "JDK" in a certain of data collection; but we believe that term "JDK" is more specific than the term "LIB" for describing "Java Programming Language"; and the 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.

IR models are the basis of ranking algorithms that are used in search engines to produce the ranked list of documents [98]. A ranking model sorts a set of documents according to their relevance to a given query [22]. For a given query, phrases were very effective and crucial in building good ranking functions with large collections [52, 88].

Relevance feedback has been used widely in the area of information retrieval to improve the ranking algorithms. It has been shown to be effective with different kinds of retrieval models [34, 66, 69, 72, 104]. The idea of relevance feedback is to involve the user in the retrieval process so as to improve the final result set. For example, a user will engage in a cyclic pattern while looking for information. At the beginning the user will formulate and submit the initial query Q, then the search engine will return a list of ranked documents R based on the similarity to

the query Q. Subsequently the user will analyse some of the documents returned. Based on that analysis some of documents D will be judged, relevance D^+ or irrelevance D^- to what user needs. The $D \subset R$, is called the relevance feedback (RF).

Based on that observation, relevance feedback is a subset of retrieved documents that have been judged by the user. It comprises of both positive and negative documents. The user's profile can be built using relevance feedback. The relevance feedback that this research used is a set of documents that have been judged by users.

Relevance Feedback: is a subset of retrieved documents that have been judged by the users (1 Relative, 0 Non-relative).

It has become essential to utilise user feedback to extract more knowledge that describes what the user meant by the query terms and to achieve the personalisation task [82]. The feedback can be implicit feedback or explicit feedback. Implicit feedback is extracted from the user's browsing history without any effort from the user, which is out of the scope of this research. Explicit feedback is the feedback given directly by the user. In other words, the search engine returns a ranked list of documents related the query, the user will check some of those documents and add his own judgements to them. The judgement can be 1 which means it is related to the user's topic of interest or 0 which means it is not related to what user needs.

2.4.2.1 Negative Relevance Feedback

Many people believe that there is plenty of negative information available and negative documents are very useful because they can help users to search for accurate information [88]. However, whether negative feedback can indeed largely improve filtering accuracy is still an open question. The existing methods of using 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 [65] based filtering models). These heuristics are 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 [7], rough set models [42] and BM25 [65]). 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, paragraphs 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++.

2.4.2.2 Rocchio Method

This model is one of the early, effective models of relevance feedback. Rocchio uses both positive and negative feedback in the training set to update user profiles in vector space models. It can be generalised as: [88, 105]

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

where U is the user profile vector, D^+ is the set of relevant documents, D^- is the set of non-relevant documents, and α and β are the empirical parameters.

Rocchio can be used to describe the features of positive documents only in the training dataset. The equation for using positive only can be simplified as:

$$U = \frac{1}{|D^+|} \sum_{\vec{d} \in D^+} \frac{\vec{d}}{||\vec{d}||}$$

Or simply set $\beta = 0$, which will ignore the negative documents.

2.5 Summary

In this chapter, the background of knowledge discovery and data mining has been discussed and the related research work regarding text mining, association analysis, text representation, information retrieval, information filtering and adaptive information filtering has been reviewed. Traditional information filtering (IF) models were developed based on a term-based user profile approach (see [45, 54, 65]). The advantage of term-based profiles is efficient computational performance as well as mature theories for term weighting, which have 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 [35, 74, 75]. Recently, language modelling approaches went beyond the term-based model that underlies BM25 by considering term dependencies in phrases (N-grams) for information retrieval [52, 88]. Although phrases are less ambiguous and more discriminative than individual terms, the likely reasons for the discouraging performance include: (*i*) phrases have inferior statistical properties to words since they have low frequency of occurrence, (*ii*) the theory of computing probabilities based on term dependencies is not practical, (*iii*) some language model-based feedback methods cannot naturally handle negative feedback, and (*iv*) 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 [21, 107]. Other promising techniques were pattern taxonomy models (PTM) [28, 90, 91] 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 in effectiveness [89]. However, two challenging issues arose with the introduction of pattern mining techniques 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 to not be suitable in the filtering stage [45]. The second issue is how to effectively use negative feedback to revise extracted features (including patterns and terms) for information filtering.

Negative feedback can be used to improve the quality of discovered features. Negative feedback can be used to revise terms that appear in both positive samples and negative samples (e.g., Rocchio based models and SVM [65] based filtering models). Other approaches review the weight of terms based on how often terms appear or do not appear in positive samples and negative samples (e.g., probabilistic models [7], and BM25 [65]).

To deal with new income training documents adaptive models have been developed to deal with this issues. In adaptive model there are two main areas of focus. The first involves updates of the user's profile to follow changes in the user's interests and this is out of the scope of this thesis. The second area involves updating the user profile to solve nonmonotonic problems.

Chapter 3

Relevance Feature Discovery

Many people believe that there is a large amount of negative information and negative feedback documents available that can be useful to help users to search for accurate information [88]. However, whether negative feedback can indeed largely improve filtering accuracy is still an open question. To solve that question, in this chapter we introduce the theoretical framework of the proposed method called Relevance Feature Discovery (RFD). The proposed method effectively uses both high-level patterns and low-level terms extracted from positive and negative feedback documents. Features extracted from negative feedback documents are used to improve the quality of extracted features from positive feedback documents.

3.1 Two Levels of Features

The proposed method is effectively using two kinds of features extracted from feedback documents. Those two kinds of features are low-level terms and high-level patterns, each of which has its own strengths and weaknesses. In this section, we introduce the two levels of features in more detail and show how to use both of them in the RFD model.

3.1.1 Low Level Features

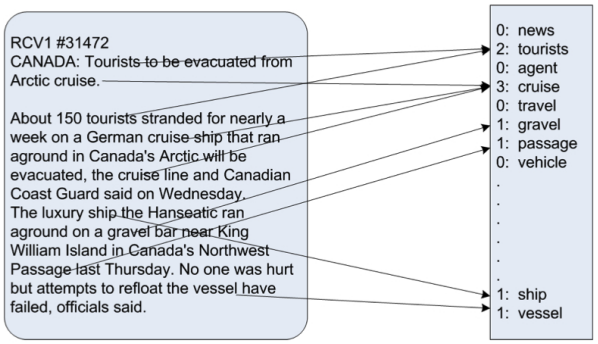

FIGURE 3.1: Document representation using bag-of-word model.

Term-based user profiles have been known as the most mature theories for term weighting to emerge over the last couple of decades in IR and machine-learning communities (see [45, 54, 65]). They have also been used widely to develop most of the traditional IF systems. A term-based model is based on the bag-of-words, which uses terms as elements and evaluates term weights based on terms' appearances or frequencies in documents. Figure 3.1 illustrates a paradigm of the bag-of-words technique. As shown in Figure 3.1 each word in the document is

retrieved and stored in a vector space with its weight. The context of this document then can be represented by these words, known as "features". For example, Rocchio-style classifiers [43], ranking SVM [60]; and BM25 for structured documents [64] are popular IF systems that use terms. They can also naturally handle both positive and negative feedback information. However, the main drawback of this scheme is that the relationship among words cannot be reflected [78]. Another problem in considering single words as features is the semantic ambiguity, which can be categorised into two categories:

- Synonyms: a word which shares the same meaning as another word (e.g. taxi and cab).
- Polysemy: a word which has two or more meanings (e.g. river "bank" and CITI "bank").

3.1.2 High Level Features

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 patterns and reducing the number of noisy patterns. As a result, patterns are less ambiguous and more discriminative than individual terms and they should give better result than terms. However, many experiments do not support this hypothesis. Likely reasons for the discouraging performance include:

- Patterns have inferior statistical properties to words since they have low frequency of occurrence.
- The theory of computing probabilities based on term dependencies is not practical.
- Some language model-based feedback methods cannot naturally handle negative feedback.
- There are large numbers of redundant and noisy low-level terms in extracted high-level patterns.

Moreover, two challenging issues have arisen with the introduction of pattern mining techniques to IF systems. The first one is how to deal with low frequency patterns because the measures used in data mining (e.g. "support" and "confidences") to learn the patterns turn out to be not suitable in the filtering stage [45]. The second issue is how to use negative feedback in pattern-based models to improve the effectiveness of the filtering process.

3.1.3 Deploying High Level Features to Low Level Features

The deploying method is used to exploit the advantages of both high-level and low-level features. The deploying method aims to represent extracted patterns from feedback documents in a term space as shown in Figure 3.2. The evaluation of low-level term supports (weights) is different from the term-based approaches in the deploying method. In the term-based approaches, the evaluation of a

given term's weight is based on its appearance in documents. In the deploying method, terms are weighted according to their appearance in discovered high-level patterns.

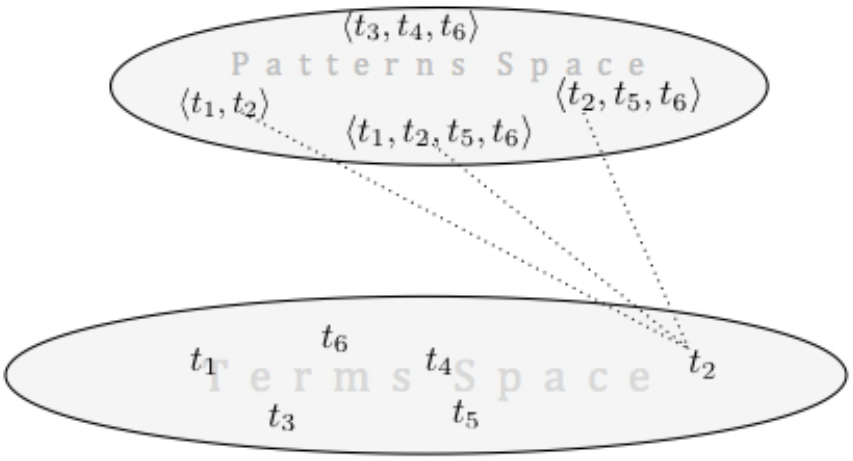

FIGURE 3.2: Deploying high-level patterns to low-level terms.

Extracting high-level patterns first and then deploying them on the low-level term space to calculate term weights would help to reduce the number of noisy terms, and give more accurate weights to terms. The obvious reason is that terms that appear in both short patterns and short pattern's super patterns would get larger weights.

3.2 Relevance Feedback

Relevance feedback has proven very effective for improving retrieval accuracy [34, 49, 66, 69, 72, 104]. The idea of relevance feedback (RF) is to involve the user in the retrieval process to improve the result set. From an initial query, the system can provide a set of results (text documents). Then, the user gives feedback to

the system by providing judgements on the retrieved documents. The judgements can be positive or negative. As a result, there are two kinds of feedback documents: positive feedback and negative feedback. Many models have shown effective performance when using positive feedback. On the other hand, some models also introduced a method of using both positive and negative feedback; for example, term-based Rocchio model. However, negative feedback has been found to be far less useful than positive feedback [51]. In this research, we are using the two kinds of features to extract high quality features from both positive and negative feedback.

3.2.1 Mining Positive Relevance Feedback

Many models have used positive feedback to improve the retrieval and filtering system. One of the recent models is the pattern taxonomy model (PTM). The PTM introduced pattern mining of the relevance feedback. In this research the PTM is used to extract knowledge from positive feedback documents. Then, a new method of using negative feedback is introduced in this thesis to improve the quality of extracted features from positive feedback documents.

3.2.1.1 Pattern Taxonomy Model

The main objective of the PTM is to extract useful patterns in text documents and use the extracted patterns to improve the effectiveness of a knowledge discovery system [89]. In the PTM, text documents are split into a set of paragraphs. Each paragraph consists of a set of words and is used as an individual transaction. To extract patterns from the transactions at the subsequent phase, the

data mining method is applied to each paragraph to extract patterns. In text documents, a sequential pattern mining algorithm (SPM) extracts a large number of patterns, but most of them are not useful. Therefore, predefined minimum support ($minimum\_sup$) is important to eliminate the less frequent patterns. In the PTM, there are two kinds of supports [89]:

- Absolute support: Absolute support is the number of occurrences of pattern P in document d in paragraphs.

$$supp_a(P, d) = |\{S|S \in d, P \sqsubseteq S\}|$$

  where S is a sequence representing a paragraph in document d.

- Relative support: is used to consider the length of the documents when extracting patterns.

$$supp_r(P, d) = \frac{supp_a(P, d)}{|dp|} \tag{3.1}$$

  where $|dp|$ is the number of paragraphs in document d.

The relative support of a pattern is used to estimate the significance of the pattern. Usually, a pattern with the same frequency will acquire the same absolute support in different document lengths. However, with the same number of occurrences, a pattern is more significant in a short document than in a long one. Using relative support to set the threshold can eliminate some patterns that occur many times in long documents and also help some patterns to pass threshold when the document is short. A sequential pattern P is called frequent sequential pattern if $supp_r(P) \geq minimum\_sup$.

TABLE 3.1: Frequent patterns and covering sets.

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

The data mining method generates a large number of sequential patterns, most of which are meaningless and redundant. In order to remove most of non-useful patterns, closed sequential pattern mining has been proposed for extracting patterns. A closed sequential pattern is defined as a frequent sequential pattern P if there exists no frequent sequential patterns P' such that $P \sqsubset P'$ and $supp_a(P) = supp_a(P')$. The relation $\sqsubset$ represents the strict part of the subsequence relation $\sqsubseteq$. For example, Table 3.1 contains ten frequent patterns; however, it includes only three closed sequential patterns: $\langle t_3, t_4, t_6\rangle$, $\langle t_1, t_2\rangle$, and $\langle t_6\rangle$.

Normally, long patterns in the taxonomy, for example pattern $\langle t_3, t_4, t_6\rangle$ in Table 3.1 are usually more specific because they have lower frequency in the documents and carry more semantic information than short patterns. The difficulty here is the lower frequency of the larger patterns. To get rid of this difficulty, the deploying method has been applied to sequential closed patterns.

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 *initial weight* ($weight_i$) in discovered patterns can be described based on the *relative support* Eq. (3.1) as follows:

$$weight_i(t, D^+) = \sum_{i=1}^{n} \sum_{t \in p \subseteq SP_i} \frac{supp_r(p, d_i)}{|p|} \tag{3.2}$$

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

After the weight of terms has been computed from the positive feedback documents, the PTM uses the following rank function to assign rank scores to every incoming document d to decide its relevance.

$$rank(d) = \sum_{t \in T} weight_i(t, D^+)\tau(t, d) \tag{3.3}$$

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

3.2.2 Mining Negative Relevance Feedback

As mentioned previously, there are two kinds of feedback documents: positive feedback and negative feedback documents. The PTM is more effective of using positive feedback only for building user profiles [41]. However, both positive and negative feedback documents are acquired for the same topic. Therefore, we believe that using both positive and negative documents can improve the quality of discovered features (user profile). In this section, we discuss the theoretical framework of using negative feedback to improve the quality of extracted features

from positive feedback documents. We call the framework a RFD model, which uses negative feedback to improve the quality of the PTM.

3.2.2.1 Hypothesis

Generally, both positive and negative feedback documents share some knowledge (features) because they are all retrieved by the same query. Using positive feedback documents can remove most of irrelevance documents [30, 62, 69]. However, not all information provided in positive documents is what users need. It is difficult to find the piece of knowledge that a user needs, standalone with no extra information in a document, as shown in Figure 3.3. As shown in the figure, the document contains more than one paragraph. The user is interested in a few lines in one paragraph. Those few lines prompt the user to judge the document as a relevant document. However, usually features are extracted from all paragraphs in the document due to the difficulty in determining what the user needs.


Java (programming language)

From Wikipedia, the free encyclopedia

"Java language" redirects here. For the Indonesian spoken language, see Javanese language.

Not to be confused with JavaScript.

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.[9][10]

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Classpath.

FIGURE 3.3: Specific knowledge required by the user.

The extra information provided by a positive feedback document can be a background or more details about what user needs. Therefore, there is an overlap between knowledge in both positive and negative documents. Based on that observation, positive documents consist of specific positive knowledge, general knowledge and noise; consequently, negative documents consist of general knowledge and noise as shown in Figure 3.4.

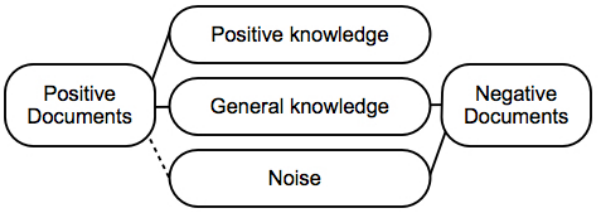

FIGURE 3.4: Relationship between both positive and negative feedback documents

Using negative feedback could help improve the quality of extracted features from positive feedback documents by identifying the three kinds of knowledge in feedback documents. The main question now is how to identify these three kinds of knowledge. To do that we take advantage of overlap between positive and negative feedback knowledge to apply the human being perspective to describe the relevance of a piece of information or knowledge.

From a human being perspective, usually people use "specificity" or "exhaustivity" to describe the relevance of a piece of knowledge such as a document, paragraph or term. For example, we believe that the term "JDK" is more specific than the term "Java" for describing "Java Programming Language"; but the term "Programming" is more general than the term "Java" since term "Programming" is also frequently used to describe all programming languages include Java, C, C++, ... etc.

Based on those observations, this research simulates a human being's perspective regarding the concept of relevance to determine the specificity of the terms. From this point of view, negative relevance feedback can be used to isolate different kinds of features extracted from feedback documents. However, using negative relevance feedback will face some challenges that need to be solved. Those challenges can be generalised in the following two questions:

- How to select constructive negative samples in order to reduce the space of negative documents? In other words, as negative is the opposite of positive then anything except what the user needs is negative. Therefore, negative documents are very diverse and they have no clear boundary.
- How to identify noisy extracted features that should be updated based on the selected negative samples?

To solve these difficult problems, this research introduces the concept of offenders which are negative documents that are likely to be classified as positive documents. It also considers how to group the extracted features into two groups: the general group and specific group, in order to accurately determine a term's weight .

3.2.2.2 Offenders Selection

Negative feedback documents have no clear boundary and have a high diversity. Therefore, using all negative feedback would lead to the increase of the boundary of noisy knowledge and the error rate. To deal with that, it is important to select negative documents that are very close to the features of positive documents

(called offenders) and keep the error rate as low as possible. An offender document is a negative document that most likely would be classified as a positive document and force the system to make a mistake. The offender documents space is illustrated in Figure 3.5. To define the offenders space; the positive feedback documents are used as a centroid of all feedback documents. Then negative feedback documents that are close to the centre are selected as offender documents.

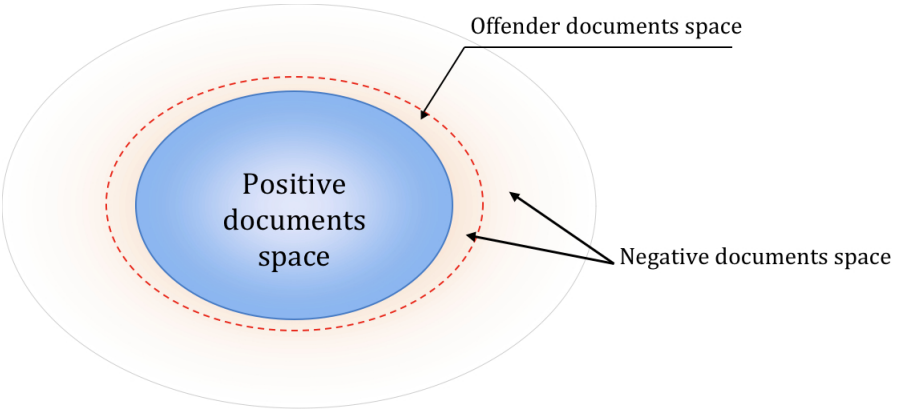

FIGURE 3.5: Offender documents in relevance feedback.

Offender documents need to be selected under the following characteristics:

- Documents that are weighted greater than 0 based on extracted features from positive feedback. In other words, documents that have some overlap information with positive feedback.
- Documents that are weighted highest based on extracted features from positive feedback.
- The number of offender documents should be less than the number of positive feedback documents in order to maintain the advantages over using positive feedback alone.

3.2.3 Feature Classification and Weight Revision

The features extracted from positive feedback may include some specific features that are specific to what the user needs, and some general features that are related to what the user needs to some extent. Some of that extra knowledge will also appear in some of the irrelevance feedback. A term's specificity describes the extent of the term to which the topic focuses on what the user wants. It is very difficult to measure specificity based on topics because a term's specificity depends on the user's perspective for their information needs [83]. Basically, the specificity of a term can be determined based on its position 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 the context of the topics users are talking about. For example, "knowledge discovery" will be a general term in the data mining community; however, it may be a specific term when talking about the broader topic of information technology.

Given a term $t \in T$, its $coverage^+(t)$ is the set of positive documents D^+ that contain t, and its $coverage^-(t)$ is the set of negative documents that contain t. Then negative feedback can be used to group the extracted features from both positive and negative feedback into three groups based on where they appear in the feedback set. 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 positive documents into the positive specific category; and the terms that are more frequently used in the negative documents into the negative specific category. The three categories in more details are:

- Category of specific positive features: which contains all features that describe what the user needs. The terms that are allocated into this group appear frequently in positive feedback documents. Features in this group should satisfy the following constraint:

$$coverage^{+}(t) > coverage^{-}(t)$$

- Category of general features: this group contains features that appear in both positive and negative feedback documents. These features describe some extent of the relevant knowledge to what the user needs.

- Category of specific negative features: which contains all features that describe some knowledge that is out of the user needs. The terms that are allocated in this group appear frequently in negative feedback documents. Features in this group should satisfy the following constraint:

$$coverage^{-}(t) > coverage^{+}(t)$$

To get the full advantage of grouping the terms into three categories, the next step is to review the weight of the terms based on their specificity to what user needs and category to which they have been assigned. There is no doubt that specific positive terms are more important for the interested topic than other groups of terms. On the other hand, the negative specific terms discuss the disbelief about the topic. Thus, the basic idea of weight revision is to increase the weights of specific positive features, and decrease the weights of specific negative features.

Although general terms may frequently appear in negative documents, they are still important for the relevance feature discovery because they describe information that is related in some way to the given topic. Increasing and decreasing the weights of a specific category will reduce the side affect of the general terms. If general features appear in negative documents, the low weight of the negative terms will reduce the document weight. On the other hand, if general features appear in positive documents the high weight of the positive terms will increase the document weight. Therefore, we believe that using the three groups of features extracted from feedback can accurately describe user information needs.

3.3 Summary

A pattern taxonomy model (PTM) has turned out to be a promising model that uses patterns to discover high-level patterns from a set of positive feedback documents. To effectively use both high-level patterns and low-level terms, discovered patterns were deployed into term space in order to calculate the initial weight of terms obtained from the patterns in the PTM. However, a large number of noisy low-level terms are deployed from high-level patterns. Also PTM is lacking in their ability to use negative feedback documents effectively.

To enhance the PTM, we present an innovative approach called Relevance Feature Discovery (RFD) for relevance feature discovery that simulates a human being's perspective about the concept of relevance, to determine the specificity of the low-level terms. It uses negative relevance feedback to improve the effectiveness of the learning model in the filtering system. The first difficulty in using negative

training documents is that negative feedback has no clear limitation and is very diverse because it can be obtained from any topic. To solve this problem, this research introduces a method to select negative documents (called offenders) that are closed to positive documents. The selected offenders are used to calculate the specificity of low-level terms. Based on the specificity score, low-level terms can be grouped into three categories: the positive specific category, general category and negative specific category. We also propose an approach to revise low-level terms based on both their appearance in the high-level patterns and their categories. We will discuss the details in the next chapter.

Chapter 4

Algorithms for The RFD Model

In the RFD model, features are discovered from a set of positive and negative documents. To effectively use both high-level patterns and low-level terms, discovered patterns are deployed into the the term space.

There are two major issues associated with the effective use of 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 search engine can return millions of documents; however, only a few documents are of interest to a Web user. Obviously, it is not efficient to use all negative documents. The second issue is how to revise the features discovered in positive documents accurately?.

This chapter discusses how to select some offender documents then use the selected documents to group low-level terms into three groups (positive specific terms, general terms and negative specific terms) based on their appearance in a feedback set.

4.1 Offender Selection

The idea of using negative relevance feedback is to define clearly the boundary of the positive knowledge (what users need) against others knowledge. However, using all negative feedback would lead to the increase of the boundary of noisy knowledge and the error rate. To reduce the space of negative feedback, only offender documents are selected in this research. An offender document is a negative document that most likely will be classified as a positive document and force the system to make a mistake.

Practically, there are three steps to select the offender documents. The first step is to extract initial features and initial weight T from positive feedback documents D^+ as shown in Section 3.2.1. The second step is to rank the negative documents D^- using the knowledge that has been extracted from positive documents as follows:

$$\begin{aligned} rank(d) = &\quad \Sigma_{t \in d \cap T} weight_i(t, D^+) \\ D^- = &\quad \{d_0, d_1, ..., d_m\} \end{aligned} \tag{4.1}$$

where, d is a negative document and D^- is the descendent ranking order for all negative documents.

The third step is to select the top-k documents as a set of offenders.

The general and noisy features would lead to an increase in the weight of the documents if they contain many of these features. The basic hypothesis is that

the relevant features should be mainly discovered from the positive documents. Therefore, the offenders are normally defined as the top-k negative documents in a ranked list of negative documents D^-. The offender documents can be defined as:

$$D_o^- = \{d_i | d_i \in D^-, rank(d_i) > 0, i < k\}$$

where, D_o^- is the list of the offender documents that have been selected from the list of negative documents D^-.

TABLE 4.1: Example of a set of closed sequential patterns extracted from positive documents.

Document	**Closed Sequential Patterns**	$supp_r$
d_1	$\langle t_1 \rangle$	1
	$\langle t_1, t_2 \rangle$	0.8
	$\langle t_3, t_4 \rangle$	0.7
d_2	$\langle t_5, t_6 \rangle$	0.95
	$\langle t_2, t_6 \rangle$	0.88
d_3	$\langle t_5 \rangle$	0.9
	$\langle t_6, t_2 \rangle$	0.2
d_4	$\langle t_1 \rangle$	1
	$\langle t_3 \rangle$	1
	$\langle t_3, t_7 \rangle$	0.65
d_5	$\langle t_2, t_6, t_4 \rangle$	1
d_6	$\langle t_3, t_5, t_7 \rangle$	1

TABLE 4.2: Deployed high-level patterns into low-level terms.

Terms	**Initial weight**
t_1	0.80
t_2	0.32
t_3	0.50
t_4	0.34
t_5	0.57
t_6	0.34
t_7	0.33

Tables 4.1, 4.2 and 4.3 provide an example to explain the selections of offenders process. We assume that, the number of feedback documents is 13 with a distribution of $|D^+| = 6$ and $|D^-| = 7$. Table 4.1 shows a set of extracted high-level patterns from positive documents. Those patterns have been deployed to low-level terms in Table 4.2 using Eq. (3.2). Table 4.3 shows a list of negative documents ranked based on the low-level terms in Table 4.2 using Eq. (4.1).

In this example, the value of $k = \frac{6}{2} = 3$, consequently $D_o = \{d_7, d_8, d_9\}$ are selected as the offender documents. Those documents will be used to extract features in order to group and review the initial weight of features extracted from D^+. We also use the *SPMining* algorithm (See Section 2.4.1.5), the same algorithm that is used to extract high-level patterns from positive documents, to extract high-level features from D_o and deploy them to low-level features.

TABLE 4.3: Example of ranking negative documents, ranked using terms in Table 4.2.

Document	**rank**
$d_7 = \{t_1, t_2, t_3, t_7\}$	1.95
$d_8 = \{t_2, t_6, t_4, t_1\}$	1.80
$d_9 = \{t_3, t_4, t_2, t_6\}$	1.50
$d_{10} = \{t_1, t_5\}$	1.37
$d_{11} = \{t_2, t_4, t_5\}$	1.23
$d_{12} = \{t_1\}$	0.80
$d_{13} = \{t_2, t_4\}$	0.66

4.2 Specificity of Low-Level Features

A feature's specificity describes the extent to which the topic focuses on what the user wants. It is clear that patterns that consist of more terms are considered to be

more specific, however they suffer from the low frequency problem. To overcome the aforementioned problem, the discovered patterns have been deployed into a hypothesis space. The deploying method solves the low frequency problem but the new list of low-level terms has lost some specificity and semantic. It is very difficult to measure the specificity of terms because a term's specificity depends on a user's perspective for their information needs [83].

In the proposed model, the specificity of low-level terms is calculated according to the appearance of features in positive and offender documents. 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 (offender) documents that contain t. Formally the coverage of a given t in positive or negative feedback documents is denoted as follows:

$$coverage^+(t) = \{d \in D^+ | t \in d\}$$

$$coverage^-(t) = \{d \in D_o^- | t \in d\}$$

Based on those definitions, the *specificity* of a given low-level term t to what a user needs, can be calculated using the following function:

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

where n is the number of positive feedback documents.

We can easily verify that the the *specificity* score satisfies the following property if $n = |D^+|$ and $|D_o| \leq \frac{|D^+|}{2}$:

$$-\frac{1}{2} \leq spe(t) \leq 1$$

Examples of calculating specificity scores for given terms $spe(t)$ are shown in Table 4.4.

TABLE 4.4: Terms' coverages and specificity score, where $n = 6$.

Term	$coverage^+(t)$	$coverage^-(t)$	$spe(t)$
t_1	2	3	$\frac{2-3}{6} = -0.167$
t_2	4	3	0.167
t_3	3	2	0.167
t_4	2	2	0
t_5	3	0	0.5
t_6	2	1	0.167
t_7	2	1	0.167

In the proposed model (RFD), the terms are grouped into three groups (specific positive terms, general terms and noisy terms) based on their specificity to the user's topic of interest. Therefore, low-level terms that appear more frequently in positive documents are classified into the positive specific category; and the terms that more frequently appear in selected negative documents are classified into the noisy (negative) category. Then terms frequently used in both positive and negative documents are general terms. Based on the above analysis, the classification rules for determining the general low-level terms G, the positive specific terms T^+, and the noisy terms T^- are presented as follows:

$$\left(\begin{array}{ll} G & = \{t \in T | \theta_1 \leq spe(t) \leq \theta_2\}, \\ T^+ & = \{t \in T | spe(t) > \theta_2\}, \ and \\ T^- & = \{t \in T | spe(t) < \theta_1\}. \end{array}\right) \tag{4.2}$$

where θ_2 is an empirical parameter, the maximum bound of the specificity for the general terms, and θ_1 is also an empirical parameter, the minimum bound of the

specificity for the general terms.

We can easily verify that the experimental parameters θ_1 and θ_2 satisfy the following properties if $k = \frac{n}{2}$:

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

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.

The classification rules and empirical parameters are illustrated in Figure 4.1. A specific score of 1 corresponds to the center of the specific positive low-level term. Thus, low-level terms closer to the center are more important ones in describing what user need.

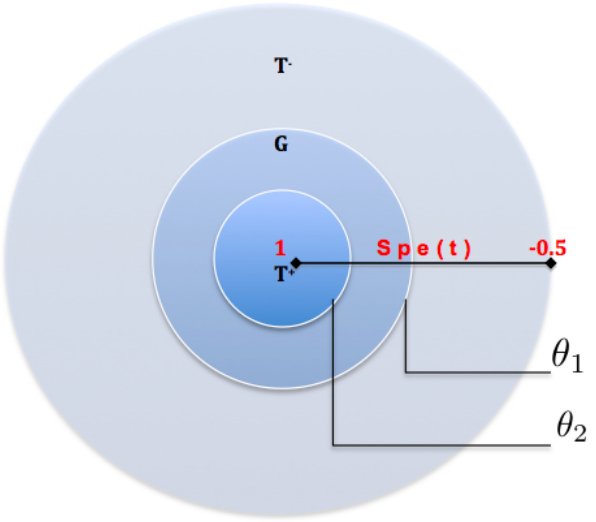

FIGURE 4.1: Grouping of low-level terms based on the specificity score.

Because positive documents may include specific terms, general terms or noise, the best way is to use the specific terms mixed with some general terms; and use

negative terms to balance the effect of general terms after reviewing the weight. This issue will be discussed in the evaluation chapter.

4.3 Weight Revision of Discovered Features

After grouping the terms into the three categories, the next step is to review the weight of the terms based on the specificity score. There is no doubt that specific positive terms are more important to the topic of interest than other groups of terms. On the other hand, the negative terms are less important. Thus, the basic idea of the weight revision is to increase the weight of the low-level terms in the specific positive group and decrease the weight of the terms in the specific negative group.

Formally, let DP^+ be the union of all discovered high-level patterns of D^+, and let DP^- be the union of all discovered high-level patterns of selected negative documents D_o^-, where a closed sequential pattern of D^+ (or D_o^-) is called a *positive pattern* (or *negative pattern*).

It is obvious 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 $t \in T^+$, an initial weight can be obtained by using the deploying method on D^+ (using high-level features).

For the terms 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_o^- as shown in the following function:

$$weight_i(t, D_o^-) = \alpha \times \sum_{i=1}^{n} \sum_{t \in p \subseteq SP_i} \frac{supp_r(p, d_i)}{|p|} \tag{4.3}$$

where $|p|$ is the number of terms in p, and α is an intuitive coefficient parameter set to -1 to give the low-level terms extracted from negative feedback documents a negative initial weight.

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

$$weight(t) = \begin{cases} weight_i(t) + weight_i(t) \times spe(t), & \text{if } t \in T^+ \\ weight_i(t), & \text{if } t \in G \\ weight_i(t) - |weight_i(t) \times spe(t)|, & \text{if } t \in T^- \end{cases} \tag{4.4}$$

where $weight_i$ is the initial weight for term t from D^+ or D_o^-.

4.4 Mining and Revision Algorithms

The process of the revision first finds features in the positive documents in the training set, including high-level positive patterns and low-level terms. It then selects some negative samples (called offenders) from the set of negative documents in the training set according to the positive features, and discovers negative patterns and terms from these using the same pattern mining technique that we used for the feature discovery in positive documents. In addition, the process reviews the initial features and obtains a revised weight function. To understand

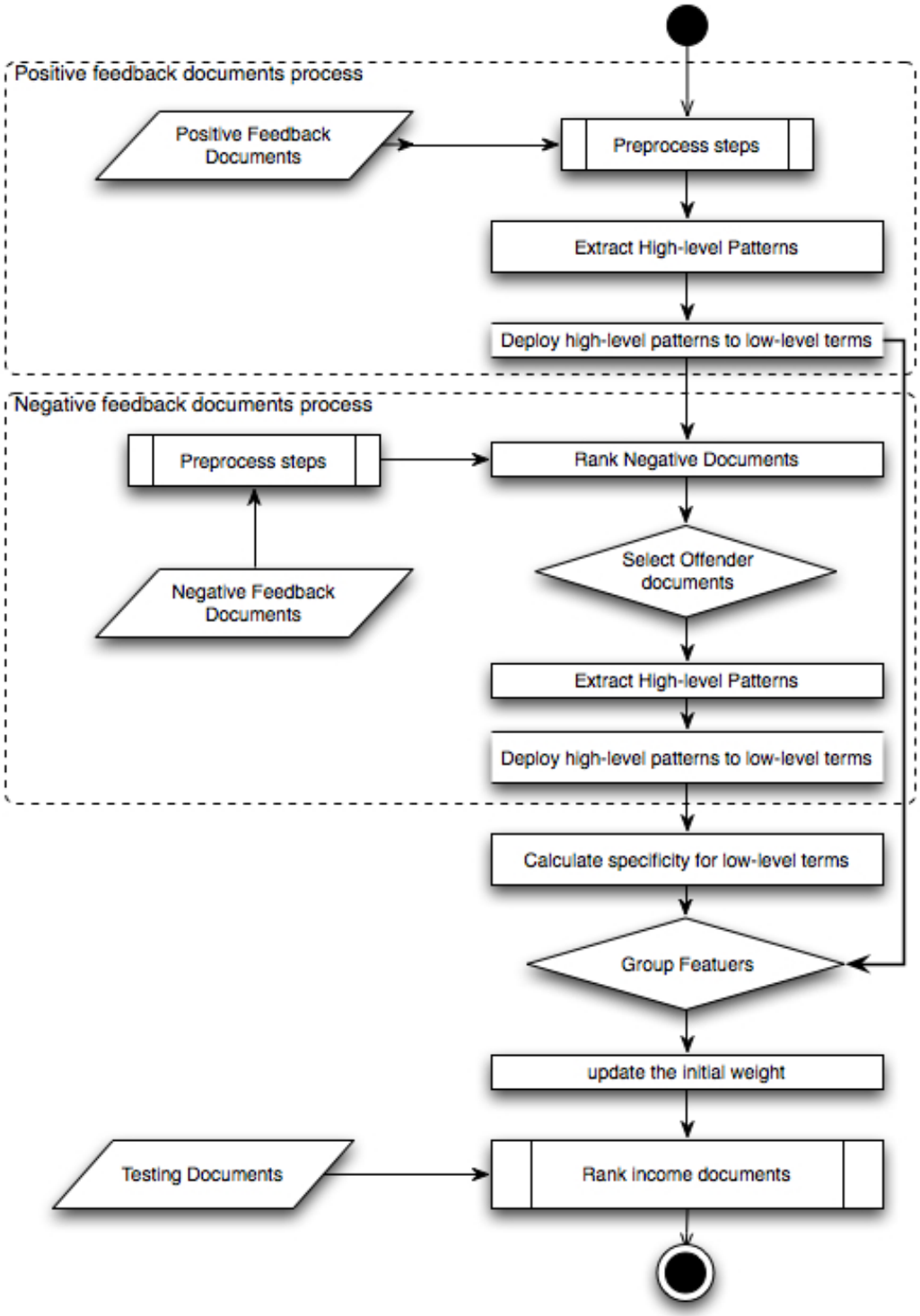

FIGURE 4.2: The process of the RFD algorithm.

HLFMining(D)

Input: A training set, $D = D^+ \cup D^-$, parameter $minmum\_sup$;
Output: extracted features $< DP^+, DP^-, T >$,
updated training set $\{D^+, D^-\}$,
and an initial term weight function, $weight_i$.

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

this process clearly, we divide this process into two algorithms: *HLFMining* and *NRevision*. The former finds high-level positive features, selects some negative samples, discovers high-level negative features, and composes the set of terms. The latter revises the term weigh function based on the high-level features and the specificity of the terms. Both algorithms are illustrated in Figure 4.2.

Algorithm *HLFMining* describes the details of high-level feature discovery. It takes a training set and a minimum support, $minmum\_sup$. It firstly extracts 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. Finally, it gives the initial weights to the terms that only appear in the negative patterns, and updates the set of terms.

The algorithm twice calls the algorithm $SPMining$: once for positive documents and once for offenders (a part of negative documents). It also takes times to calculate weights of terms that appear in the discovered patterns, and sorts the negative documents, taking $O(mlog^m)$, where $m = |D^-|$. Compared with the time complexity of the algorithm $SPMining$, the time spent calculating weights 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 $D = \{D^+, D_o^-\}$, using *HLFMining(D)*, we can obtain the extracted features $< T, DP^+, DP^- >$, and an initial weight function w over T. Given the experimental parameters θ_1 and θ_2, algorithm *NRevision* describes the details of revising the weights of the terms based on their specificity and distributions in both positive and negative patterns.

Step 1 initialises 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 first calculates the specificity for all terms, and then determines positive specific terms, general terms and negative specific terms based on the classification rules as presented in Eq. (4.2). Finally, it updates the initial weights of terms using the function *weight* Eq. (4.4). 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

***NRevision*( )**

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

Output: A term function *weight*.

Method:

$G = \emptyset, T^+ = \emptyset, T^- = \emptyset, n = |D^+|$;
foreach $t \in T$ **do** { //term partition
  $spe(t) = \frac{|\{d|d \in D^+, t \in d\}| - |\{d|d \in D_o^-, t \in d\}|}{n}$;
  if $spe(t) > \theta_2$
    then $T^+ = T^+ \cup \{t\}$;
    else if $spe(t) < \theta_1$
      then $T^- = T^- \cup \{t\}$;
      else $G = G \cup \{t\}$; }
foreach $t \in T^+$ **do**
  $weight(t) = weight_i(t) + weight_i(t) * spe(t)$;
foreach $t \in T^-$ **do**
  $weight(t) = weight_i(t) - |weight_i(t) * spe(t)|$;

average size of the documents, $|D_o^-|$ is the number of offenders and $|D_o^-| \leq n$. Therefore, the time complexity of this algorithm is $O(|T| \times |d| \times n)$.

4.5 Summary

This chapter presents the details of the proposed approach for relevance feature discovery. The system first extracts high-level patterns from positive feedback documents. To effectively use the extracted features; the high-level patterns are deployed into low-level terms. In contrast to the term-based approaches, the initial weights of the low-level terms in the RFD model are based on the support of high-level patterns. To improve the quality of extracted features from

positive feedback, we propose a method to select negative documents (called offenders) that are close to what user needs. Then both high-level patterns and low-level terms are extracted from the offenders. The specificity score to what the user needs is calculated for each low-level term. Based on the specificity score function, the low-level terms are classified into three categories: Specific positive category, General category and Specific negative category) . In order to improve the performance of the system, the initial weights of the low-level terms are revised based on their category and their specificity. The evaluation results will be presented in chapter 6.

Chapter 5

Adaptive Relevance Feature Discovery

It has been proven that Relevance Feedback (RF) is very effective for improving retrieval accuracy [34, 49, 66, 68, 72, 104]. In this chapter we extend the RFD model to be an adaptive model. The new adaptive model is call Adaptive Relevance Features Discovery (ARFD). It automatically updates the system's base knowledge on a sliding window over positive and negative feedback to solve a non-monotonic problem efficiently. For example, given the system's limited number of training documents about the term "Agent", the IF systems may return information objects such as "Intelligent Agent", "Property Agent", or "Software Agent". However, when more information about the user's actual information needs is obtained later on, the system can determine that the user is only interested in "Software Agent" [32].

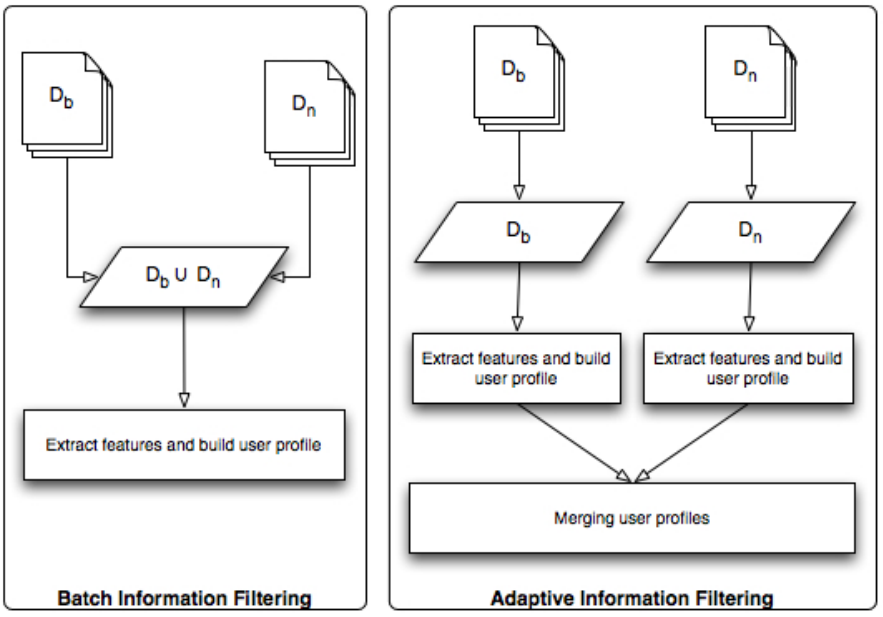

FIGURE 5.1: Adaptive and batch information filtering.

5.1 Adaptive Information Filtering

The idea of updating the system with new training documents is to provide the system with new knowledge to follow user need changes over time and improve the effectiveness of the filtering process. Generally, in adaptive information filtering, the system begins with small samples of training documents to solve the cold start problem. The cold start problem concerns the issue that the system cannot draw any inferences for users or items about which it has not yet gathered sufficient information. After solving the cold start problem, the user provides more feedback about what they want as they go along [101]. The new feedback is a list of documents that have been judged (relevant 1, irrelevant 0) by the user.

There are two ways to update the system with the new feedback documents. The easy way is to combine new feedback documents with the existing documents,

then train the system again from beginning (Batch Filtering). The alternative is to extract knowledge from the new training documents, then efficiently and effectively feed it into the system (Adaptive Filtering). Figure 5.1 shows the process of updating system knowledge using the two different methods: batch filtering and adaptive filtering. This research focuses on updating the system in an adaptive manner by extracting knowledge from some of the new feedback documents that may contain new knowledge different to what the system already has. Then, the new knowledge is merged with the existing knowledge to solve nonmonotonic problem efficiently.

5.2 Adaptive Relevance Feature Discovery

In this research we divide the feedback documents into two groups:

1. The initial training set with which the system starts (called basic feedback documents D_b), and

2. The new feedback set D_n. The sliding window has been used to mange new training documents D_n.

The size of the window is set to be 25 documents consisting of positive and negative training documents, $D_n = D_n^+ \cup D_n^-$.

To achieve the adaptive process, the ARFD model has been proposed for relevance features discovery. The architecture of the ARFD model is illustrated in Figure 5.2. The system works in five steps, which are generalised as follows:

1. Evaluate new feedback documents D_n using available knowledge extracted from D_b.

2. Select the documents D_s that contain new knowledge, where $D_s \subseteq D_n$.

3. Extract new features knowledge from D_s using RFD.

4. Merge the new knowledge with the base knowledge (the existing knowledge).

5. Test the new merged knowledge by using D_n. If the new merged profile gives better results than the old profile, then the system keeps the new knowledge; otherwise, it reverts back to the old knowledge.

5.2.1 Evaluate New Feedback Documents

Generally, the system starts with some knowledge (called the Base Knowledge) extracted from the initial feedback set $D_b = D_b^+ \cup D_b^-$ using the RFD model (See Chapter 3 and 4). Then the system will receive new feedback documents. The new feedback can be grouped into two main categories; The first category contains documents that are similar to the old feedback documents. The second category contains documents that are different to the old feedback documents, which may include new knowledge about user interests. This research will focus on the second category where the users provide more information about their interests for solving the nonmonotonic problem. The new feedback may overlap with the current knowledge that the system has. Based on that we can interpret the two categories of new feedback documents as follows:

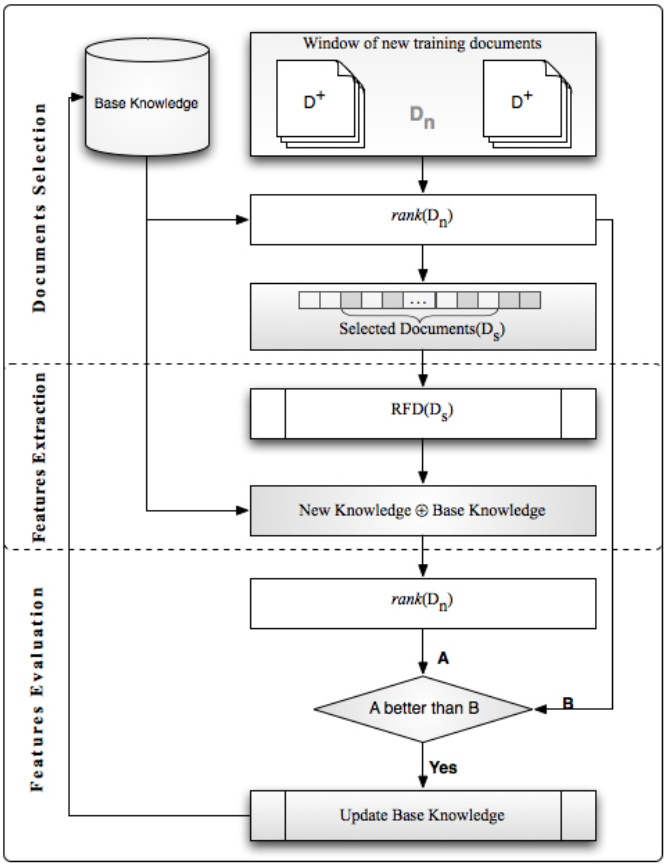

FIGURE 5.2: ARFD system architecture.

1. The first category contains knowledge that the system has clearly already learnt; therefore, they are redundant documents.

2. The second category could contain new knowledge that has not yet been learnt by the system yet.

To avoid overloaded knowledge and reduce the storage and the time complexity, the new approach uses only new training documents that can introduce new knowledge to the system. To select those documents, the new training documents

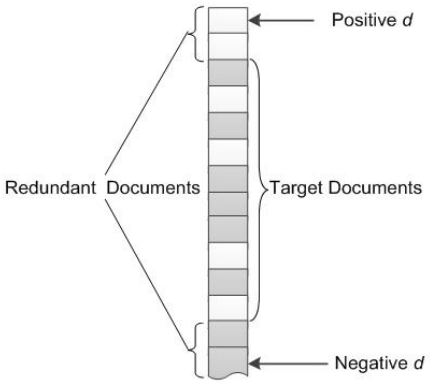

FIGURE 5.3: Document selection.

should be evaluated and ranked using the base knowledge of the system. The rank function is described as follows:

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

where $weight(t) = weight(t, D_b)$; and $\tau(t, d) = 1$ if $t \in d$; otherwise $\tau(t, d) = 0$.

5.2.2 Document Selection

After ranking the new documents, the most relevant documents are located in the top of the list and most irrelevant documents are at the bottom of the list, as shown in Figure 5.3. These documents are classified correctly because the system has the knowledge to deal with those documents. These documents are viewed as redundant documents because they contain similar knowledge to that already existing in the system. To remove this redundancy, we use a high precision threshold, (e.g 95%). We start from the top of the ranking list, calculate the precision for each document (cut-off=1) and stop when the precision is less than

TABLE 5.1: Comparison of all pattern (phrase) based methods on all topics.

Document	Human Judgement	Precision for d^+	Precision for d^+
d_1	1	1	:
d_2	1	1	:
d_3	1	1	:
d_4	0	0.75	:
:	:	:	:
d_{23}	1	:	0.67
d_{24}	0	:	1
d_{25}	0	:	1

the threshold. The precision for positive documents can be calculated using the following function for each cut-off point:

$$precision^+ = \frac{TP}{TP + FP}$$

where TP (True Positive) is the number of documents that the system correctly identifies as positives; FP (False Positive) is the number of documents that the system falsely identifies as positives.

The same idea applies for the negative documents, starting at the end of the list, where we expect most of negative documents to be. The precision for negative documents can be calculated using a similar function for each cut-off point:

$$precision^- = \frac{TN}{TN + FN}$$

where TN (True Negative) is the number of documents that the system correctly identifies as negative; FN (False Negative) is the number of documents that the system falsely identifies as negative.

The target documents ranked between 95% precision of positive and 95% precision

SNTSelect(D_n, T)

Input: A new training set, $D_n = D_n^+ \cup D_n^-$,
parameter min_sup;
extracted low level features T from D_b;
Output: extracted features from D_s,
Method:
1: let $m = |D_n|$, $TP = 0$, $TN = 0$, $s = 1$, $e = m$;
2: **foreach** $d \in D_n$**do**
3: $rank(d) = \Sigma_{t \in d \cap T} weight(t, D_b)$;
4: let $D_n = \{d_1, d_2, ..., d_m\}$ in descending ranking order,
5: **for** $i = 1$ **to** m;
if $(d_i \in D_n^+)$ **then** TP++;
else TN++;
$precision^+ = \frac{TP}{(TP+TN)}$;
if $precision^+ < 0.95$ **then** $s = i$: **exit**;
6: $TP = 0$, $TN = 0$;
7: **for** $i = 1$ **to** m;
if $(d_{m+1-i} \in D_n^-)$ **then** TN++;
else TP++;
$precession^- = \frac{TN}{(TP+TN)}$;
if $precession^- > 0.95$ **then** $e = m + 1 - i$: **exit**;
8: $D_s = \{d_i | d_i \in D_n, s \leq i < e\}$;

of negative will be selected (D_s) to be used for updating the system, because those documents contain some new knowledge that system does not already have. Table 5.1, contains 25 documents ranked using knowledge extracted from D_b. All documents from d_4 to d_{23} will be selected as D_s. Figure 5.3 also illustrates the concept of document selection.

Algorithm *SNTSelect* describes the details of selecting some documents D_s from new training documents D_n. We assume that the initial features, $< T, DP^+, DP^- >$, have been extracted from the initial training set D_b. The algorithm takes the new training set D_n and the extracted features from D_b. It first ranks the documents D_n in step 2 to step 4. After that, it removes the redundant documents and

selects the target documents D_s that will be used to update the initial knowledge. Algorithm *SNTSelect* takes time to rank and select some of the training documents, $O(mlog^m)$, where m is the number of new training documents.

5.2.3 Knowledge Extraction

Generally, the system starts with some knowledge (called Base Knowledge) extracted from the initial feedback set $D_b = D_b^+ \cup D_b^-$ using the RFD model (See Chapter 3 and 4). The features T_b and the knowledge (functions) extracted from the base knowledge set D_b can be described as follows:

$$\langle T_b, weight^b, weight_i^b, coverage_b^+, coverage_b^-, |D_b^+| \rangle$$

When the system receives new feedback $D_n = D_n^+ \cup D_n^-$, it then selects a new training set D_s from the incoming feedback D_n, such that $D_s \subseteq D_n$. To use the selected documents from the new feedback documents, each document in D_s is pre-processed and split into paragraphs. Therefore, a given document d yields a set of paragraphs $PS(d)$. Then the RFD algorithm is used to extract features from both training data sets without applying the revision process that is discussed in algorithm *NRevision()* in chapter 4.

In more detail, all closed sequential patterns DP_s^+ in D_s^+ are extracted used the *SPMining* algorithm [39]. The result of *SPMining* is a list of high-level patterns with *support*. To get low-level terms and weights from the high-level patterns, the

deploying function (Chapter 3) is applied to these patterns. The initial weight $weight_i$ for each term is calculated based on the support of closed sequential patterns in D_s^+ using Eq. (3.2). Let $T_n = \{t_{s1}, t_{s2}, \ldots, t_{sm}\}$ be a set of terms with initial weight $weight_i(t_s)$ which are deployed from DP_s^+ extracted from D_s^+. Then, a cover set for given term $t_s \in DP^+$ is represented as:

$$coverset^+(t_s) = \{p | p \in D_s^+, t \in p\}$$

Some of the negative documents are also selected from D_s^- (called offender D_{so}) [39] to update and revise the terms extracted from D_s^+. Similar to features extracted from D_s^+, closed sequential patterns DP_s^- are extracted from the selected negative documents in D_{s_o}. Then the deploying function is applied to DP_s^- to evaluate the initial term weights. The cover set for a given term t_s in offender documents is denoted as:

$$coverset^-(t_s) = \{p | p \in D_{s_o}^-, t \in p\}$$

If the term t_s appears only in D_s^+ then $coverset^-(t_s) = \emptyset$, and also if the term appears only in D_s^- then $coverset^+(t_s) = \emptyset$.

Based on the above steps, the features T_n and the knowledge (functions) in the new feedback set D_n can be described as follows:

$$\langle T_n, weight^n, weight_i^n, coverage_n^+, coverage_n^-, |D_s^+| \rangle$$

5.2.4 Knowledge Merging

Both features and knowledge in the initial training set and the new training set will be merged as the final result of the adaptive process. The final features $T' = T_b \cup T_n$; $|D'^{+}| = |D_b^{+}| + |D_s^{+}|$; and for term $t' \in (T_n \cap T_b)$, its functions are updated as follows:

$$\begin{pmatrix} weight'_i(t') = & weight^b_i(t') + weight^n_i(t') \\ coverage^{+}(t') = & coverage^{+}_b(t') \cup coverage^{+}_n(t') \\ coverage^{-}(t') = & coverage^{-}_b(t') \cup coverage^{-}_n(t') \\ |D'^{+}| = & |D_s^{+}| + |D_b^{+}| \end{pmatrix} \tag{5.1}$$

The functions of term t that appear only in one group T_n or T_b will be the same in the new merged knowledge. On the other hand, for the terms that appear in both T_n and T_b, the term weights are merged as shown in Eq. 5.1.

The revised Algorithm *NRevision()* (see Section 4.4) in the RFD model is used to revise the weights after merging. The specificity score of a given term t' can be calculated using the $spe(t')$ function in the RFD model as:

$$spe(t') = \frac{|coverage^{+}(t')| - |coverage^{-}(t')|}{D'^{+}}$$

Each term t' can be allocated in one of the three groups T'^{+}, G' and T'^{-}.

Finally, the initial weights of terms are revised according to the following principles: increase the weights of the positive specific terms T'^{+}, decrease the weights of the negative specific termsT'^{-}, and keep the weight of general terms G' as is.

The details are described as follows:

$$weight(t') = \begin{cases} weight_i(t') + weight_i(t') \times spe(t'), & \text{if } t \in T'^{+} \\ weight_i(t'), & \text{if } t \in G' \\ weight_i(t') - |weight_i(t') \times spe(t')|, & \text{if } t \in T'^{-} \end{cases}$$

where $weight_i$ is the initial weight from D_b and D_s.

The time complexity of knowledge extraction and the merging technique are mainly decided by the number of selected new training documents. Therefore, the time complexity of this algorithm is $O(|T| \times |d| \times n)$. In the merging process for each term t, it takes $O(|d| \times (n + |D_s|)) = O(|d| \times n)$ for evaluating $spe(t)$, where $|d|$ is the average size of the documents, $|D_s|$ is the number of offenders and $|D_s| \leq n$. Therefore, the time complexity of this algorithm is $O(|T| \times |d| \times n)$.

5.2.5 Feature Evaluation and Decision Making

The combine knowledge expected to improve the system judgement. However, that is not the case due to the quality of the new training documents and the quality of extracted knowledge. Moreover, number of documents each group of training D_b and D_s would also affect the result of the merged knowledge. To control those factors and garnet that the new training documents would lead to better result the new merged knowledge need to be tested.

To test the effectiveness of the new merged knowledge extracted from D_s and D_b, the set of new feedback documents D_n is used as a testing set. If the merged result is better than using only knowledge extracted from D_b, the new combined

knowledge will be used. Otherwise, the new knowledge will be ignored and the system will revert to using the base knowledge only.

5.3 Summary

In summary, the adaptive model (ARFD) is built based on the RFD model that uses a data mining technique to extract patterns from relevance feedback. The aim of this method is to update the system using new feedback documents efficiently and effectively to solve the nonmonotonic problem. The process of the ARFD model consists of the following steps: (1) Mining features and functions from the initial (or the base) training set D_b; (2) Selecting some documents D_s that contains new knowledge to the system from a new training set D_n to remove some redundant documents; (3) Mining features and functions from the target documents D_s; and (4) merging these features and the functions discovered from both the initial training set and the selected documents. The objective of this process is to learn new knowledge in the new feedback set in order to refine the discovered knowledge in the initial training set. The evaluation results will be presented in chapter 6.

Chapter 6

Evaluation

Two main hypotheses have been proposed in this research. These two hypotheses are:

- Some negative feedback documents are useful to assist in removing the noise in discovered features from positive feedback documents.
- Not all new feedback documents are useful for updating the existing learning model. Moreover, not all extracted knowledge from useful documents can easily harmonise with the existing knowledge.

To evaluate the proposed hypotheses, this chapter discusses the testing environment including the dataset, baseline models, and evaluation methods. For the first hypothesis, it reports the results and the discussions for the following main points: the proposed model is significant compared to the baseline models based on effectiveness; the effectiveness of using different negative feedback documents is different; and the classifications are useful to reduce the noise 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 and negative documents.

For the second hypothesis, the proposed model is significant compared with the baseline models in efficiency and effectiveness. Also, it provides more results and discussion about document selection and knowledge merging. The proposed model is also a supervised approach that needs a training set including both positive documents and negative documents.

6.1 Dataset

The most frequently used collection for experiments in the information filtering area is the Reuters dataset. During the last decade, several versions of Reuters corpora have been released. The latest version of these common data collections has been chosen for this experiment, which is the Reuters Corpus Volume 1 (RCV 1) [70]. The RCV1 dataset contains about 100 topics. Documents in each topic are divided into two groups, training and testing. The structure of the RCV1 dataset is shown in Figure 6.1.

The RCV1 dataset contains 100 topics and each topic contains a reasonable number of documents with relevance judgement both in the training and testing examples. Figure 6.2 shows the number of documents in each topic and in both the training and testing categories. The 100 topics in the RCV1 dataset were

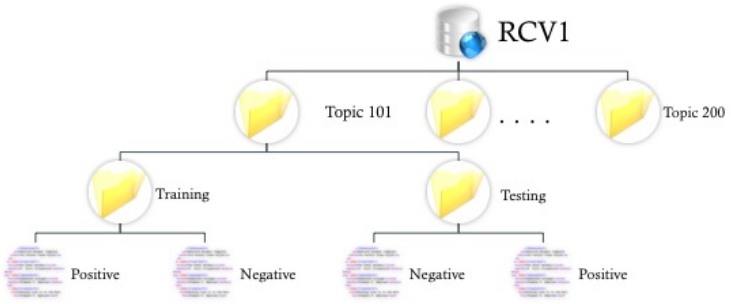

FIGURE 6.1: Training and testing documents.

collected from English language news stories produced by Reuters journalists between August 20, 1996, and August 19, 1997, in a total of 806,791 documents. However, only the first fifty of these documents were categorised by humans and researchers; the rest were categorised by intersecting two Reuters topic categories [5]. As a result, the scores on the first 50 topics (assessing) are more reliable than the last 50 topics categorised by machine.

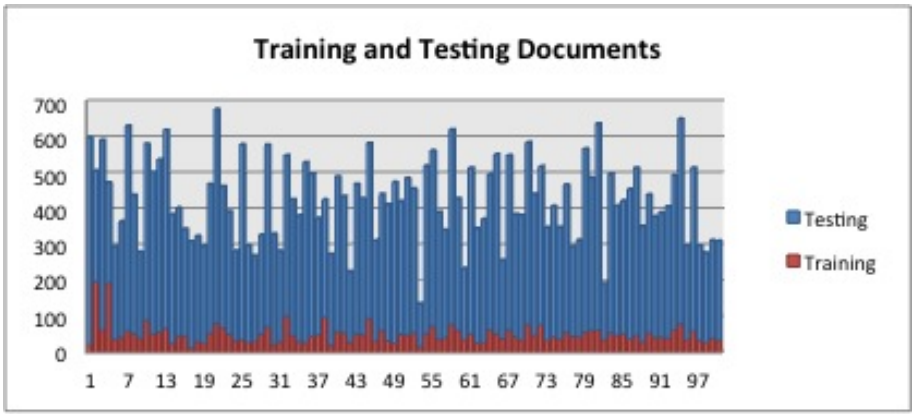

FIGURE 6.2: Training and testing documents.

Each RCV1 topic was divided into two sets: training and test. Relevance judgements have been provided for each document on each topic. The training set is

composed of a total of 5,127 news stories with dates up to and including September 30, 1996, and the test set contains 37,556 news stories from the rest of the collection. Figure 6.3 and 6.4 show the number of relevant and irrelevant documents in both the training and testing sets.

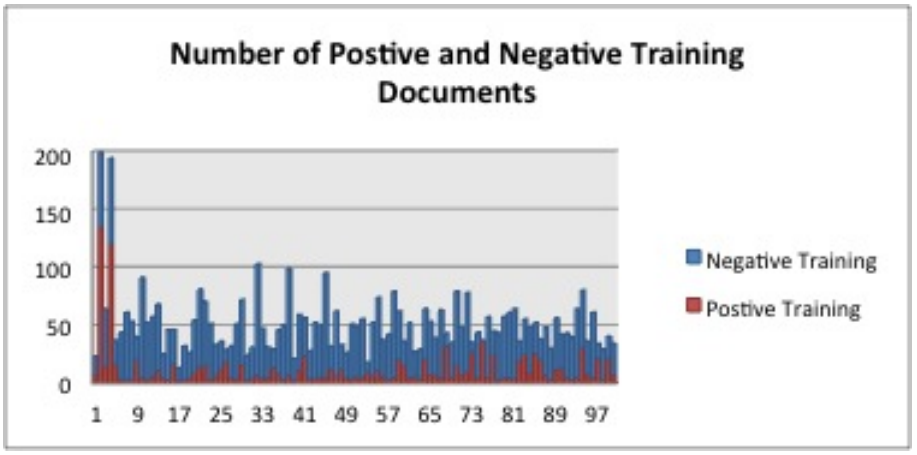

FIGURE 6.3: Number of positive and negative training documents.

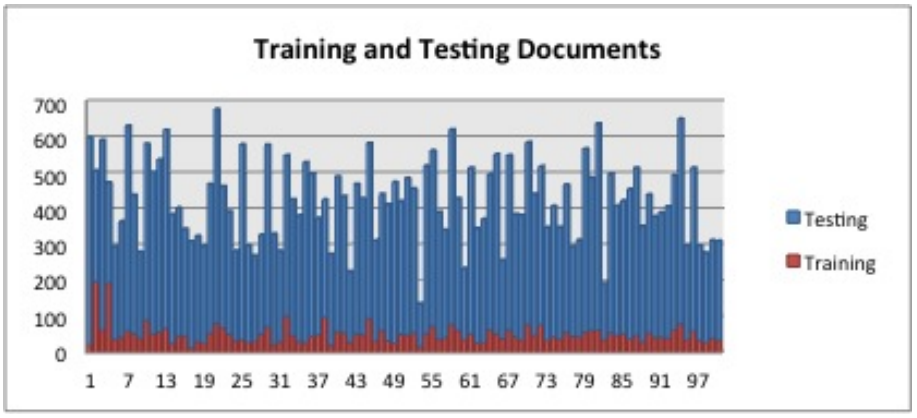

FIGURE 6.4: Number of positive and negative testing documents.

Stories in both sets are classified as either positive or negative. "Positive" means that the story is relevant to the assigned topic; otherwise, the word "negative" will be shown. In our experiments, we selected all 100 TREC topics (from topic

101 to topic 200). Further details regarding the RCV1 dataset can be found in [70]. The RCV1 dataset is distributed on two CDs and contains about 810,000 English language stories. It requires about 3.7 GB for storage if all the files are uncompressed.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="26642" id="root" date="1996-09-01" xml:lang="en">
<title>INDIA: At least 44 dead as vessel capsizes in India.</title>
<headline>At least 44 dead as vessel capsizes in India.</headline>
<dateline>NEW DELHI 1996-09-01</dateline>
<text>
<p>At least 44 people were feared drowned when their vessel capsized in the
Nagavalli river in the southern state of Andhra Pradesh, the United News of
India said on Sunday.</p>
<p>It quoted official sources as saying the boat was carrying some 50 people,
 mainly tribespeople, when it sank on Saturday.</p>
<p>Six people swam to safety, it said.</p>
</text>
<copyright>(c) Reuters Limited 1996</copyright>
<metadata>
<codes class="bip:countries:1.0">
  <code code="INDIA">
    <editdetail attribution="Reuters BIP Coding Group"
    action="confirmed" date="1996-09-01"/>
  </code>
</codes>
<codes class="bip:industries:1.0">
  <code code="I74000">
    <editdetail attribution="Reuters BIP Coding Group"
    action="confirmed" date="1996-09-01"/>
  </code>
</codes>
<codes class="bip:topics:1.0">
  <code code="GCAT">
    <editdetail attribution="Reuters BIP Coding Group"
    action="confirmed" date="1996-09-01"/>
  </code>
  <code code="GDIS">
    <editdetail attribution="Reuters BIP Coding Group"
    action="confirmed" date="1996-09-01"/>
  </code>
</codes>
<dc element="dc.date.created" value="1996-09-01"/>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1996-09-01"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="NEW DELHI"/>
<dc element="dc.creator.location.country.name" value="INDIA"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

FIGURE 6.5: An XML document in RCV1 dataset.

The documents in the RCV1 dataset are tagged using XML format for easy access and parsing. An example of an RCV1 document is illustrated in Figure 6.5. Each

document is identified by a unique item ID and corresponds to a title in the field marked by the tag $< title >$. The main content of the story is in a distinct $< text >$ field consisting of one or several paragraphs. Each paragraph is enclosed by the XML tag $< p >$. In this research experiment, both the "title" and "text" fields are used and each paragraph (i.e. content in $< p >$) in the "text" field is viewed as a transaction in a document. Moreover, we treat the content in the "title" field in the document as an additional paragraph (i.e. transaction).

Preprocessing Steps

In preprocessing, redundant terms need to be eliminated before the documents can be interpreted by the system. Since RCV1 documents are all in XML format, there are many fields enclosed by tags including $< title >$, $< headline >$, $< dateline >$, $< text >$, $< copyright >$ and $< metadata >$. In our experiments, the fields we chose in each document are $< title >$ and $< text >$. The content of the rest of the fields is discarded. In an RCV1 document, each $< text >$ field contains several paragraphs enclosed by tag $< p >$. For implementing RFD and ARFD, we treat each paragraph as a transaction, as well as the content in the field $< title >$ which is viewed as an extra paragraph because of the rich information it contains. The next process is to apply stopword removal and word stemming. In stopword removal, function words and non-informative terms are removed according to a given stopword list. Stopwords can be defined as frequently occurring, insignificant words that help construct sentences, i.e., articles, prepositions, and conjunctions. Simple stopwords are the words that appear in all English documents. Common english stopwords include [48]:

$$a, about, an, are, is, as, at, be, by, for, from, how, in, of, on, or, that,$$
$$the, these, this, to, was, what, when, who, will, with$$

For word stemming, the Porter algorithm is used for suffix stripping [59].

6.2 Evaluation Methods

It is important to measure the performance of an information system. As with many of the IF systems utilised in this research, we are only interested in one class "positive". The positive class is the class that matches the information required by a particular user. The rest of the documents are classified as negative. Accuracy is not a suitable measure in some cases, such as when the data set is comprised of 99% positive cases. The system can achieve 99% accuracy by simply classifying all data sets as positive [48] . In the area of information filtering, precision p and recall r are suitable because the measure of precision and complete classification is based on the positive class.

TABLE 6.1: Confusion matrix of a classifier.

		Human judgement	
		Yes	**No**
System judgement	**Yes**	TP	FP
	No	FN	TN

Precision is the fraction of retrieved documents that are relevant to the topic, and recall is the fraction of relevant documents that have been retrieved. For a binary classification problem the judgement can be defined through the confusion

matrix as depicted in Table 6.1. According to the definition in this table, the precision and recall are denoted by the following formulas:

$$p = \frac{TP}{TP + FP},$$
$$r = \frac{TP}{TP + FN},$$

where TP (True Positive) is the number of documents the system correctly identifies as positives; FP (False Positive) is the number of documents the system falsely identifies as positives; FN (False Negative) is the number of relevant documents the system fails to identify, and TN (True Negative) is the number of documents that the system correctly identifies as negative.

Comparing two classifiers based on precision and recall is very difficult. In the test set, precision can be high, but recall may be very low and vice versa. The F_{score} (also called the F_{1score}) is often used to compare models in the IF and IR area. The F_{score} is denoted by the following formula:

$$F_{score} = (1 + \beta^2)\frac{p \times r}{\beta^2 p + r};$$

The value of β depends on how concerned we are about false negatives versus false positives, such as intelligent filtering versus spam filtering, which conventionally equals 1. The F_{score} is the harmonic mean of precision and recall, which tends to be closer to the smaller of the two. Thus, the F_{score} will be high if both precision and recall are high.

Some people argue that the precision values of the top K documents are more

important than the precision values of all the positive documents because the top K documents are the most likely documents that the user is going to read. The precision of the first K of retrieved documents top-K is also adopted in this thesis due to the fact that most users would focus more on the first few dozen returned documents. The precision of top-K returned documents refers to the relative value of relevant documents in the first Kreturned documents. The value of K used in the experiments in this thesis is 20.

There is also another measure called precision and recall breakeven point (b/p), which is usually applied in the IF and IR areas. The value of the breakeven point is calculated when precision and recall are equal. Both the b/p and F_{score} are single-value measures and are used to reflect performance across all the documents. The following example gives a clear explanation of this measure.

Example: Assume that we have a list of 20 ranked documents. The first one presents the highest rank and the last one presents the lowest rank. Assume that the test set has 10 positive sample documents:

$$
\begin{array}{lll}
At\ rank1: & P = 1/1 = 100\% & r = 1/10 = 10\% \\
At\ rank2: & P = 2/2 = 100\% & r = 2/10 = 20\% \\
\vdots & \vdots & \vdots \\
At\ rank9: & P = 6/9 = 66.7\% & r = 6/10 = 60\% \\
At\ rank10: & P = 7/10 = 70\% & r = 7/10 = 70\% \\
\vdots & \vdots & \vdots
\end{array}
$$

The interpolated average precision (IAP) measure compares the performance of different systems by averaging the precision from 11 standard recall levels. The

11-point measure is used in our comparison tables indicating the first value of 11 points, where recall is equal to zero that means the smallest recall value; e.g $\frac{1}{TP+FN}$. The IAP was calculated by measuring the precision of each relevant document first and averaging the precision across all topics.

The effectiveness is measured by five different means: The Mean Average Precision (MAP), average precision of the top 20 documents, F_1 measure, *breakeven point* (b/p), and Interpolated Average Precision (IAP) on *11-points*. The larger the score of the *top-20*, MAP, b/p, IAP and F_1-measure, the better the system performs. The *11-points* measure is also used to compare the performance of different systems by averaging precisions from 11 standard recall levels (i.e., recall=0.0, 0.1, ..., 1.0).

Statistical method is also used to analyse the experimental results. The *t-test* assesses whether the means of two groups are statistically different from each other or not. The paired two-tailed *t-test* is used in this thesis. 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 are not 0). The test statistic is t with $N - 1$ degrees of freedom (df), where N is the sample size of group. 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.

6.3 Baseline Models and Setting

We need to compare the proposed models with other models including the state-of-the-art models. To do that, baseline models are grouped 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. We do not select language models because they are naturally not available for negative feedback.

In the experiments, the proposed approach is also divided into two stages. The first stage uses only positive patterns in the training set. The model, called PTM, discovers sequential closed patterns from positive documents, deploys discovered patterns on their terms using the following function (see section 2.4.1):

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

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

Then, testing documents will be ranked used the following function:

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

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

The second stage used 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 Chapter 3 and 4. 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 $minmum\_sup = 0.2$ (relative support) for all models that use patterns. The n-gram model uses *3-grams*.

For the adaptive model, all models implemented, trained and tested use the same training dataset on the same machine. However, the training documents in all baseline models are a combination of the base training dataset and the new one. All the systems combined all training set and use it as one training set at the beginning to perform batch filtering.

Our approach is called Adaptive Relevance Features Discovery (ARFD). This approach is based on the RFD model and adds the ability to update the extracted knowledge adaptively. To get the experimental result, the system at the beginning uses an initial training dataset. Later on a new feedback documents will be provided to the system.

In RFD and ARFD, the minimum support $minmum\_sup = 0.2$ (relative support), $\theta_1 = 0.2$ and $\theta_2 = 0.3$. The size of the window for the new training documents for ARFD is set to 25 documents. For each topic, we also choose 150 terms in the positive documents, based on *tf*idf* values for all term-based baseline models.

6.3.1 Rocchio Model

The Rocchio algorithm [69] has been widely adopted in the areas of text categorisation 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:

$$\alpha \frac{1}{|D^+|} \sum_{\vec{d} \in D^+} \frac{\vec{d}}{||\vec{d}||} - \beta \frac{1}{|D^-|} \sum_{\vec{d} \in D^-} \frac{\vec{d}}{||\vec{d}||} \tag{6.1}$$

There are two recommendations for setting parameters α and β in the Rocchio model [9, 29]: $\alpha = 16$ and $\beta = 4$; and $\alpha = \beta = 1.0$. We have tested both recommendations on the RCV1 dataset and found $\alpha = \beta = 1.0$ was the best one. Therefore, we set $\alpha = \beta = 1.0$ in Eq. (6.1).

6.3.2 BM25 Model

BM25 [65] is one of the 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 thesis).

6.3.3 SVM Model

SVM achieved the best performance on the Reuters-21578 data collection for document classification [99]. SVM was also used in [11] as a classifier for pseudo-relevance feedback, where multiple features were utilised, such as term distribution, co-occurrence with query terms, and term proximity. 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 labelled positive (negative). $\alpha_i \, \epsilon \, \Re$ is the weight of the sample x_i and satisfies the constraint:

$$\forall_i : \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{6.2}$$

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 of each negative document based on Eq. (6.2). 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

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

6.4 Experimental Setting

All the experiments reported in this thesis were conducted on a PC equipped with an Intel Pentium IV 3.0GHz CPU and 1,024 MB memory running a Windows XP operating system. The application of the RFD-based IF system and ARFD was coded using Java programming language with J2SDK version 1.4.2 as the development environment. The data collection was acquired from a licensed CD from the TREC organisation and used in our experiments without any modification, although we found some errors and duplicates in the data. The information of relevance judgement for each topic in training and test datasets was also derived from files directly downloaded from the TREC Web site 2.

The value of minimum support used for association rule mining in the experiments was set to 0.2 according to the system optimisation. For the reason

of consistency, we used the same minimum support in all related mining algorithms. The influence of various settings on minimum support is a well-studied issue which has been widely investigated in the data mining literature [53]. Thus in our experiment, we did not focus on this coefficient. Moreover, during the recursive loop of proposed mining algorithms, the loop would stop and exit if no more patterns were being found. However, in some cases (e.g. topic r193 and r199) the recursive loop did not seem to stop as some documents in these topics contain a large number of long patterns. The longest pattern we found was 15, using SCPM and SPM mining algorithms. Therefore, for non-sequential pattern mining algorithms (i.e. NSPM and NSCPM), the maximum length of pattern we searched for was set to 15 and the loop could exit after such long patterns were found no matter whether or not any longer candidate could be generated.

6.5 Evaluation Procedures

In order to evaluate the proposed RFD and ARFD models, we apply them to the practical information filtering task. There are three kinds of information filtering: adaptive filtering, batch filtering, and routing filtering. On one hand, the RFD model uses batch filtering to build user profiles, but uses routing filtering to test the incoming documents. Routing filtering retrieves a ranked list of documents rather than making boolean decisions. On the other hand, the ARFD model uses adaptive filtering to build the user profile, but uses routing filtering to test the income documents. Adaptive filtering has the ability to use new feedback documents to update user profiles as long as the system is running. In this thesis, routing filtering is implemented and the performance of the model is evaluated

based on the ranked documents. The choice of the routing task avoided the need for threshold tuning, which is beyond our focus in this research work.

We evaluated the proposed models using all 100 TREC topics (r101 to r200) in the experiments. TREC provides two sets of documents for each topic for training and test purposes. Table 6.2 and Table 6.3 provide the related statistical information for training and test datasets respectively. All the documents in these two sets are processed both in the phase of profile learning and document evaluation.

TABLE 6.2: Number of relevant documents($\#r$) and total number of documents($\#d$) by each topic in the RCV1 training dataset.

Topic	#d	#r	Topic	#d	#r	Topic	#d	#r	Topic	#d	#r
r101	23	7	r126	29	19	r151	49	6	r176	57	5
r102	199	135	r127	32	5	r152	55	5	r177	45	25
r103	64	14	r128	51	4	r153	18	10	r178	43	3
r104	194	120	r129	72	17	r154	52	6	r179	57	5
r105	37	16	r130	24	3	r155	74	11	r180	61	5
r106	44	4	r131	31	4	r156	37	6	r181	64	4
r107	61	3	r132	103	7	r157	42	3	r182	36	19
r108	53	3	r133	47	5	r158	79	5	r183	55	25
r109	40	20	r134	31	5	r159	62	21	r184	48	9
r110	91	5	r135	29	14	r160	36	15	r185	52	26
r111	52	3	r136	46	8	r161	52	5	r186	38	20
r112	57	6	r137	50	3	r162	27	6	r187	48	7
r113	68	12	r138	98	7	r163	29	4	r188	30	3
r114	25	5	r139	21	3	r164	64	21	r189	56	12
r115	46	3	r140	59	11	r165	53	7	r190	42	13
r116	46	16	r141	56	24	r166	39	8	r191	43	5
r117	13	3	r142	28	4	r167	63	5	r192	40	3
r118	32	3	r143	52	4	r168	43	32	r193	64	5
r119	26	4	r144	50	6	r169	35	5	r194	80	31
r120	54	9	r145	95	5	r170	79	16	r195	36	8
r121	81	14	r146	32	13	r171	48	7	r196	61	5
r122	70	15	r147	62	6	r172	78	10	r197	34	22
r123	51	3	r148	33	12	r173	35	27	r198	29	3
r124	33	6	r149	26	5	r174	44	5	r199	40	21
r125	36	12	r150	51	4	r175	37	37	r200	34	7

TABLE 6.3: Number of relevant documents(#r) and total number of documents(#d) by each topic in the RCV1 test dataset.

Topic	#d	#r	Topic	#d	#r	Topic	#d	#r	Topic	#d	#r
r101	577	307	r126	270	172	r151	437	22	r176	411	37
r102	308	159	r127	238	42	r152	402	41	r177	250	61
r103	528	61	r128	276	33	r153	118	37	r178	271	47
r104	279	94	r129	507	57	r154	469	39	r179	510	32
r105	258	50	r130	307	16	r155	489	63	r180	426	72
r106	321	31	r131	252	74	r156	354	72	r181	574	25
r107	571	37	r132	446	22	r157	300	37	r182	157	32
r108	386	15	r133	380	28	r158	542	45	r183	443	139
r109	240	74	r134	351	67	r159	367	97	r184	361	13
r110	491	31	r135	501	337	r160	199	54	r185	371	184
r111	451	15	r136	452	67	r161	463	47	r186	417	264
r112	481	20	r137	325	9	r162	319	81	r187	467	31
r113	552	70	r138	328	44	r163	343	122	r188	322	36
r114	361	62	r139	253	17	r164	432	182	r189	384	76
r115	357	63	r140	432	67	r165	499	52	r190	337	85
r116	298	87	r141	379	82	r166	219	17	r191	347	18
r117	297	32	r142	198	24	r167	486	40	r192	367	29
r118	293	14	r143	417	23	r168	342	269	r193	430	16
r119	271	40	r144	380	55	r169	348	35	r194	571	187
r120	415	158	r145	488	27	r170	507	73	r195	263	37
r121	597	84	r146	280	111	r171	394	68	r196	453	50
r122	393	51	r147	380	34	r172	441	41	r197	264	144
r123	342	17	r148	380	228	r173	314	226	r198	249	18
r124	250	33	r149	449	57	r174	364	82	r199	272	116
r125	544	132	r150	371	54	r175	312	312	r200	277	86

6.6 RFD Evaluation

6.6.1 RFD Evaluation Procedures

As each document is transferred into the desired format, one of the mining methods is selected to find dedicated patterns in the phase of pattern discovery. These patterns are then passed through the subsequent deploying and evolving processes

to generate the representative concept (e.g. deployed pattern set), which is used to represent the set of positive documents. Then some offenders are selected from negative documents based on the knowledge extracted from the positive documents. The selected negative documents are put through the process again by discovering patterns and applying deploying methods to them. The extracted features from offenders are used to group and revise the weight of features extracted from positive and offender documents. In the test phase, each document in the test set is evaluated to examine the performance of the proposed models. In summary, the steps required for the whole evaluation procedure in RFD are briefly listed as follows:

1. The system starts with one of the RCV1 topics and retrieves the related information with regards to the training set, such as the file name list and the number of documents.

2. Each document is preprocessed with word stemming and stopwords removal and transformed into a set of transactions based on its document structure (paragraphs).

3. The system selects one of the pattern discovery algorithms to extract high-level patterns from positive documents (PrefixSpan).

4. Discovered high-level patterns are deployed over a terms space (low-level terms) using the deploying method.

5. Extracted features from positive feedback documents are used to select offenders from negative documents.

6. Pattern discovery algorithms are also used to extract high-level patterns from offenders.

7. The discovered high-level patterns in offenders are deployed over low-level terms using the proposed deploying methods.

8. Specificity scores are calculated for each feature extracted from positive or offender documents.

9. Group low-level terms according to their specificity score into three groups: Specific positive, General and Specific negative.

10. The weight of low-level terms is revised based on the group to which the terms belong and their specificity score.

11. The result of the filtering tasks is evaluated.

6.6.2 Overall Evaluation Result

All documents in the dataset are split into paragraphs. Each paragraph is used as a transaction. The result of document indexing is a set of transactions and each transaction consists of a vector of stemmed terms. The next step is to find frequent patterns using pattern discovery algorithms. As mentioned in Chapter 2 and 3, data mining approaches including association rule mining, frequent sequential pattern mining, closed pattern mining, itemset mining, and closed itemset mining are adopted and applied to the text mining tasks. By splitting each document into several transactions (i.e. paragraphs), we can use these mining

TABLE 6.4: List of methods used for RFD evaluation.

Method	Description	Algorithm
RFD	Relevance Feature Discovery Proposed method	HLFMining, NRevision Section 4.4
PTM Deploying	Pattern taxonomy model	PTM Section 2.4.1.5
Sequential ptns.	Data mining method using frequent sequential patterns	SPM Section 6.6.2
Sequential closed ptns.	Data mining method using frequent sequential closed patterns	SCPM Section 6.6.2
Freq. itemsets	Data mining method using frequent itemsets	NSPM Section 6.6.2
Freq. closed itemsets	Data mining method using frequent closed itemsets	NSCPM Section 6.6.2
nGram	nGram method with $n = 3$	3Gram Section 6.6.2
Rocchio	Rocchio method	Section 6.3 $\alpha = 1, \beta = 0$
BM25	Probabilistic method	Section 6.3 $\eta = 0.5$
SVM	Support vector machines method	Section 6.3 $b = 0$

methods to find frequent patterns from the textual documents. Five pattern discovery methods which have been implemented in the experiments are explained briefly as follows:

- SPM: Finding sequential patterns using the algorithm *SPMining* with skipping of the first line in the algorithm.
- SCPM: Finding sequential closed patterns using the algorithm *SPMining*.
- NSPM: Finding non-sequential patterns using the algorithm NSPMining.

- NSCPM: Finding non-sequential closed patterns using algorithm NSPMining with a closed pattern mining scheme.
- nGram: Finding all sequential patterns, whose lengths do not exceed "n", using the *SPMining* algorithm.

Note that the *minmum_sup* is set to 0.2 for all mining methods, which means a pattern is frequent if it appears in n paragraphs (including title field) in a document containing m transactions (paragraphs + title), such that $n/m \geq 0.2$. For comparison reasons, the value of *minmum_sup* will stay the same for all approaches. A comparison of these methods with the RFD model is revealed as follows.

Table 6.4 shows a list of all methods used for evaluating the RFD model. RFD is first compared with n-grams and other pattern-based models, especially PTM, the best 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.

The results of overall comparisons between RFD, n-grams and other pattern-based models in all assessing topics are presented in Table 6.5, 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. Then, using some

TABLE 6.5: Comparison of RFD with all pattern-based methods on all topics, where %*chg* is the percentage change over the best model (PTM).

Method	*top-20*	*MAP*	$F_{\beta=1}$	*b/p*	*IAP*
RFD	**0.557**	**0.493**	**0.470**	**0.472**	**0.513**
PTM Deploying	0.496	0.444	0.439	0.430	0.464
Seq Closed Ptns	0.406	0.364	0.390	0.353	0.392
Seq Patterns	0.401	0.361	0.385	0.343	0.384
Freq Patterns	0.412	0.361	0.386	0.352	0.384
Freq Closed Ptns	0.428	0.361	0.385	0.346	0.387
n-Gram	0.401	0.361	0.386	0.342	0.384
%*chg*	**+9.75**	**+11.18**	**+6.92**	**+12.30**	**+10.44**
Average of percentage change over all 5 measure is 10.12%					

of the negative (offender) feedback to revise deployed features in the RFD model improved the result significantly by about 10.12% on average compared with the PTM that used only positive feedback documents.

The result 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. Moreover, it has been proven that using two levels (high-level patterns and low-level terms) of features is giving better result than used one of them only. Finally, it is clear that using both positive and some of the negative feedback (offenders) in the RFD model to revise low-level terms can significantly improve of the effectiveness of the filtering tasks in pattern-based models. As shown in Table 6.5, RFD using both positive feedback and offenders achieves excellent performance with 10.12% (max 12.30% and min 6.92%) in percentage change on average for all five measures in all assessing topics.

The proposed method RFD is compared with term-based baseline models including Rocchio, BM25, and SVM. The experimental results on all assessing topics

TABLE 6.6: Comparison results of all models in all assessing topics, where *%chg* is the percentage change over the best model (Rocchio).

	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%**
Average of percentage change over all 5 measure is 13.42%					

are reported in Table 6.6 with the percentage changes compared with results from the other best models.

As shown in Table 6.6, the proposed new model RFD has achieved the best performance results for the assessing topics. The table shows that the average percentage of improvement over the standard measures is 13.42% with a maximum of 17.51% and minimum of 9.25% compared with the best result in term-based approaches (Rocchio).

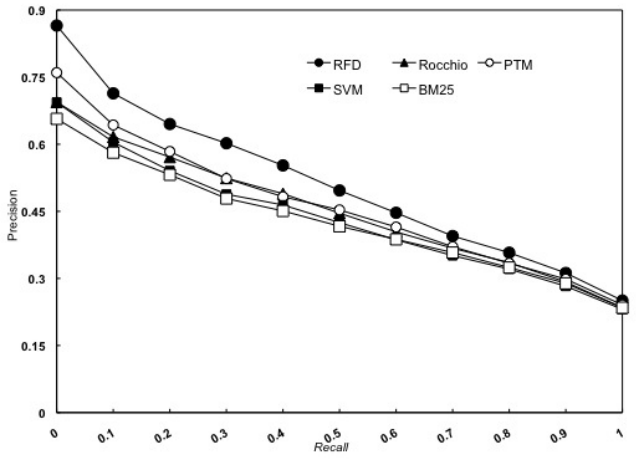

FIGURE 6.6: Comparison the results in all assessing topics.

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 6.6; and the $p - values$ of the *t-test* show that the proposed model RFD is extremely statistically significant as shown in Table 6.7. Therefore, we conclude that using some of negative feedback (offenders) to group and to revise the low-level features extracted from positive and offenders feedback is an exciting achievement for the discovery of relevance features in text documents.

TABLE 6.7: t-test p-values for all models comparing with the 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

Table 6.8 shows a comparison result between the proposed model and other baseline models in the last 50 topics starting from topic 151 to topic 200. The results in the table indicate that the proposed model are not improved comparing with some of the baseline models. The most likely reason is that the benchmark judgement of those topics is not accurate. As mentioned previously, the topics in the RCV1 dataset are split into two groups:

- The first group comprises topics 100 to 150, known as the assessing topics. These topics were evaluated by human beings.
- The second group incorporates topic 151 to topic 200. These topics were evaluated by a machine.

The results using the first group (assessing topics) are more reliable than the results using the second group that judged by some algorithms. A model can get a better result using the second group of topics if it uses a model that is close to the model used by the machine.

TABLE 6.8: Comparison results of all models from topic 151 to topic 200.

	top-20	*MAP*	$F_{\beta=1}$	*b/p*	*IAP*
RFD	**0.593**	**0.536**	**0.483**	**0.501**	**0.549**
Rocchio	0.593	0.542	0.490	0.507	0.558
SVM	0.581	0.533	0.485	0.496	0.548
BM25	0.575	0.503	0.471	0.462	0.524

6.6.2.1 RFD in TREC 2010 Relevance Feedback Track

The third year of the TREC relevance feedback (RF) track aimed to examine what makes an individual document good or bad for feedback by focusing on single document relevance feedback as the track's main task [14]. QUT E-Discovery Lab participated in this TREC using the RFD model. In TREC 2010, the RFD model was recognised and recommended, along with three other models, by the TREC organisers to be one of the baseline models in this area.

The dataset used in TREC 2010 is the ClueWeb09 dataset. The ClueWeb09 dataset was created by the Language Technologies Institute at Carnegie Mellon University to support research on information retrieval and related human language technologies. It consists of more than 1 billion web pages in 10 languages. However, only category B from the ClueWeb09 dataset was used at TREC relevance feedback in 2010. Category B contains the first 50 million English pages in the ClueWeb09. The size of category B is 1.5 TB.

The basic user scenario is that a user has submitted a short (title only in TREC parlance) query to a retrieval system that has returned one or more documents to the user. The user then identifies a relevant document and submits positive feedback on this document. Based upon this document and the original query, the system re-ranks the list of documents given to the user. This new list is then evaluated for both accuracy (top documents are relevant) and completeness (all relevant documents are retrieved, not just those that have the same aspect as the known document). Like almost all relevance feedback tasks, we assume the user needs several or many relevant documents.

The track investigated this scenario by providing groups of 100 topics for which we already had some known relevant and non-relevant documents. For each topic, we selected five documents to be used for single document relevance feedback. Groups were to treat this document as an example of a relevant document and submit five result lists for each topic. Groups also submitted baseline runs that utilised no relevance information. In addition, groups had the option to supply their opinion, in the form of a relative ranking, of how well the five feedback documents would perform.

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

suites, caroma toilet, etc.). However, it is difficult to determine which one reflects the user's information need. Based on that observation, we developed a model to work as follows:

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

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

3. The top 10 documents were selected as the positive feedback and the bottom 10 as negative feedback from the result of step 2. This pseudo-relevance feedback was used to update user profiles.

4. The pseudo-relevance feedback went into the RFD model to generate three feature sets (the positive specific terms, general terms, and negative specific terms). The three feature sets were used to re-rank the 15000 documents.

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

More details about the RFD model in TREC2010 can be found in [2].

6.6.3 Specificity of Patterns

We think most of the patterns that consist of more terms are considered to be more specific, however they suffer from the low frequency problem. To overcome

the aforementioned problem, the discovered patterns have been deploying into a term space [45, 90]. The deploying method solves the patterns' low frequency problem but the new list of low-level terms has lost some specificity and semantic.

TABLE 6.9: Pattern and term analysis results for specificity and exhaustive purposes.

Average number of extracted features			Average number of patterns used			Average length of patterns used		
DP^+	T^+	G	T^+	G	$(T^+ + G)$	DP^+	T^+	G
201.78	23.54	22.36	20.94	20.08	39.86	1.460	2.55	1.462

Table 6.9 shows that, the average length of extracted patterns from positive documents is about 1.460 for 201 patterns, and most of them have been considered to be noisy patterns. Whereas 23.54 positive terms appear only in 20.49 patterns with average length of 2.55. Comparing the average length of the patterns for both positive and general groups, it becomes clear that the average length of the patterns that contain positive terms (2.55) is greater than the average length of the patterns that contain general terms (1.462). The above analysis shows that the longer the pattern is the more specific; which matches our general thinking.

6.6.4 Offender Selection Evaluation and Discussion

The first difficulty of using negative training documents is that negative feedback has no clear limitation and can be obtained from any topic, which increases the diversity of negative feedback. To solve this difficult problem, we use the concept of offenders which are negative documents that are likely to be classified as positive documents. Steps required to select offenders are briefly listed below:

1. Extract features from positive feedback.
2. Rank negative feedback documents using features extracted from positive feedback.
3. Eliminate any documents that have weight equal to 0.
4. Select top K documents as offenders.

The objectives of the RFD model is to extract high quality features that can represent what users need. Therefore, positive feedback is more important than negative feedback. As a result, the value of K should be less than the number of positive feedback documents to retain the advantage of positive group. The K value in the RFD model is set to $\frac{|D^+|}{2}$. A sensitivity study to investigate the selection of offenders is presented below.

6.6.4.1 Offender Selection Discussion

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

Technically, selecting a large number of negative documents as offenders will weaken the importance of the positive T^+ and G general group of low-level terms.

TABLE 6.10: Statistical information for the RFD ($\theta_1 = 0.2$,$\theta_2 = 0.3$) with different values of K in assessing topics.

K	Average number of training d			Average number of terms			Average weight of terms			*top-20*	*MAP*	$F_{\beta=1}$
	Positive	Negative	Offenders	$\#T^+$	$\#G$	$\#T^-$	$w(T^+)$	$w(G)$	$w(T^-)$			
RFD ($\lvert D^+\rvert/2$)	12.78	41.3	6.54	23.54	22.36	231.78	4.159	1.400	-0.551	0.557	0.493	0.470
$\lvert D^+\rvert$	12.78	41.3	10.18	20.78	20.74	280.18	3.060	1.270	-2.964	0.537	0.463	0.450
$\lvert D^-\rvert$	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 6.10 shows the average numbers of positive documents, negative documents and offenders in the training sets for all the assessing topics by using different values of K. From that table, we can see that the larger the value of K, the fewer the number of features or the lower the weights in T^+ and G. Moreover, it's easy to see the significant increase of terms and weights in T^- in the table. Another advantage of offender selection is to reduce the space of negative relevance feedback. Table 6.10 clearly illustrates that only $15.8\% = \frac{6.54}{41.3}$ of the negative documents are selected as offenders in the RFD model.

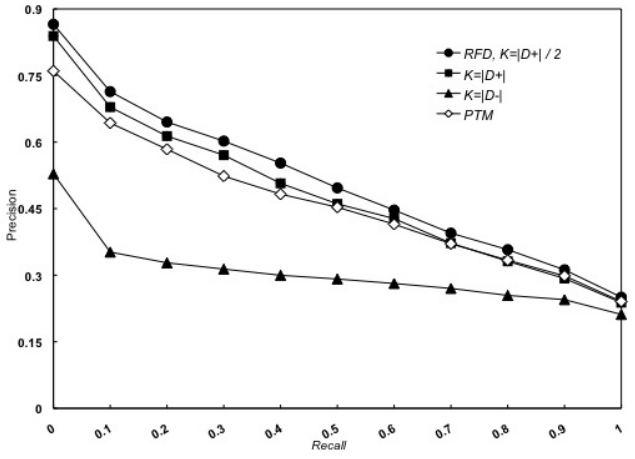

FIGURE 6.7: Comparison results using different values of K in the RCV1 dataset.

Figure 6.7 shows the performance for different settings of K for offender selection for all assessing topics. If we use all negative documents, the performance is

largely even worse than when using the PTM. 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.

6.6.5 Classification Rules and Weight Revision Evaluation and Discussion

The specificity function describes how a term is related to a topic of interest. Then, two thresholds θ_1 and θ_2 are used to decide the category of low-level terms. The objectives of grouping the terms into three categories (Specific positive, General and Specific negative) are to isolate noisy terms and reduce the side affect of general terms. Reducing the side affects of noise and general terms can be done by reviewing the weight of features in each group according to the specificity score and the category of the low-level terms.

6.6.5.1 Classification Rules

TABLE 6.11: Results of using different values for θ_1 and θ_2 in 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

Low-level terms are grouped based on the specificity function and the classification rules. Two thresholds are used to decide the category of the terms based on

specificity score. Table 6.11 shows some possible values for the two thresholds θ_2 and θ_1. Classification of the low-level terms into three categories gives a distinct advantage when later reviewing the weight of terms.

TABLE 6.12: Statistical information of the RFD and PTM in assessing topics.

Extracted Terms used RFD			Average $weight(t)$ before revision			Average $weight(t)$ after revision			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

Table 6.12 shows the average number of terms extracted by the proposed model and the PTM. The average number of terms for each topic that the PTM extracted from positive documents was 156.9, and all those terms were assigned to one group. However, there are many noisy terms in the group which impose restrictions on the effectiveness. After using the proposed model to group the terms into three categories, the number of terms in both the specific positive category and the general category is reduced to $45.9 = 23.54 + 22.36$, that is, only 29.25% are retained in RFD model. Table 6.12 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 6.12, the percentage of general terms is $48.71\% = \frac{22.36}{22.36+23.54}$ compared to 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 users want. To further reduce the side affects of using general terms in relevance feature discovery, the proposed method adds negative specific terms into the low-level features. After using offenders, Table 6.12 shows that about $120.78 = (23.54 + 22.36 + 231.78) - 156.9$ in average new negative

features T^-, were added into the user profile, and they are assigned negative weights.

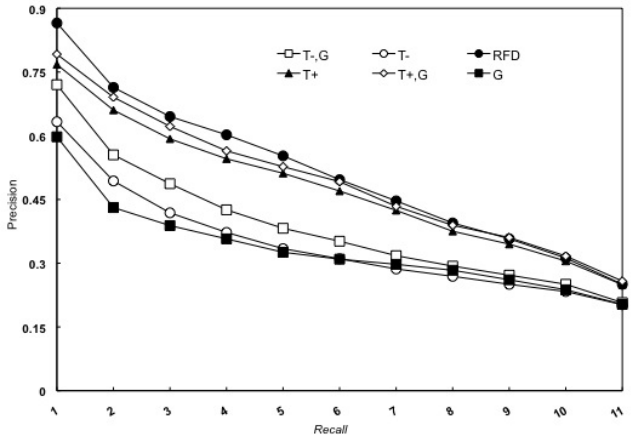

FIGURE 6.8: Comparison results using different groups of terms in all assessing topics.

Based on the above analysis and the hypothesis, the proposed model using the general group of knowledge G should give the worst result. This is because general knowledge is the knowledge that is likely to confuse the system and cause bad result. Using negative group knowledge T^- should give a poor result but no worse than the general group because most of the low-level terms in the negative group carry negative weights. The positive group of knowledge T^+ should give the best result but no better than using all groups of knowledge. Figure 6.8 shows the result of the RFD model using different groups of low-level terms. The result supports the objectives of classification rules that we proposed.

6.6.5.2 Weight Revision

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 6.12, before revision, $66.99\% = \frac{2.84211}{1.40038+2.84211}$ of total weightings are distributed to the specific positive terms, and 33.01% of total weightings are distributed to general terms. After revision, the weight distribution changes to $25.19\% = \frac{1.40038}{1.40038+4.15839}$ for general terms and 74.81% for specific positive terms. A sample of term weight distribution trends before and after revision is shown in Figure 6.9.

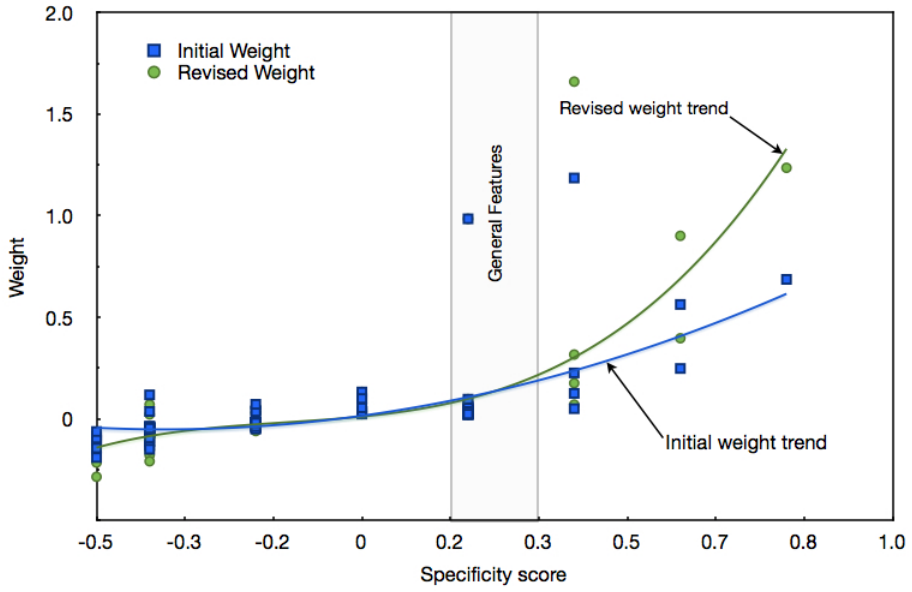

FIGURE 6.9: Sample of weight distribution before and after review among the extracted terms.

There are many terms which are moved to the negative specific category, for example $111 = 156.9 - (23.54 + 22.36)$ in Table 6.12. In order to reduce the

side affects of those terms, the proposed model reduces the weight of the low-level terms in the negative specific category T^-. The terms in the negative specific category are assigned on average a weight of -0.55127 after revision. In this way, these negative specific terms could reduce the side affects of general terms more if both general terms and negative specific terms appear in negative documents, because now only $16.96\% = \frac{1.40038-0.55127}{4.15839+1.40038-0.55127}$ of total weightings could be distributed to the general terms, considering that positive specific terms are assigned on average a weight of 4.15839 and general terms on average are weighted at $1.40038 - 0.55127$.

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 noise. The experimental results show that we can choose the same number of positive specific terms and general terms, and assign larger weights to the positive specific terms.

6.6.6 Summary

Negative relevance feedback is very useful for information filtering. The proposed Relevance Feature Discovery approach shows that negative feedback can largely improve filtering accuracy. It introduces a method to select negative documents (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. From this perspective, it presents an algorithm to revise extracted features. Compared with the pattern-based models and the state-of-the-art models, the results of experiments

on the RCV1 collection demonstrate that the effectiveness of information filtering can be significantly improved by the proposed new approach. This research provides a promising methodology for evaluating term weights based on discovered patterns (rather than documents) in both positive and negative relevance feedback. Results supports the objectives of this research.

6.7 ARFD Evaluation

6.7.1 ARFD Evaluation Procedures

The RFD model is the base model of the ARFD model. As a result, most of the steps in the RFD model are also applied in the ARFD model. However, ARFD uses two sets of feedback documents: the first is the basic feedback dataset provided by the RCV1 dataset called D_b. The second is a sliding window selected randomly from testing documents in RCV1 dataset, which are then used as training documents, called D_n. The steps required for the whole evaluation procedure in ARFD are listed as follows:

- The training set provided by the RCV1 dataset that is also used in the RFD model is called D_b. The same steps used in the RFD model are implemented into D_b. The final result of this step is a list of low-level terms with weights assigned.
- A new training set D_n is a list of documents selected randomly from the testing set in the RCV1 dataset. The set of new training documents D_n

is removed from the testing set. The steps to use the D_n dataset in the ARFD model are briefly listed as follows:

1. Evaluate D_n documents using features extracted from D_b.
2. Select documents D_s that contain new knowledge to the system, where $D_s \subseteq D_n$.
3. Apply the RFD Model to the selected documents D_s.
4. Merge the two sets of features extracted from D_b and D_s.
5. Evaluate the merged result using D_b as a testing set. Then make a decision whether to use the merged knowledge or just use knowledge extracted from D_b alone.

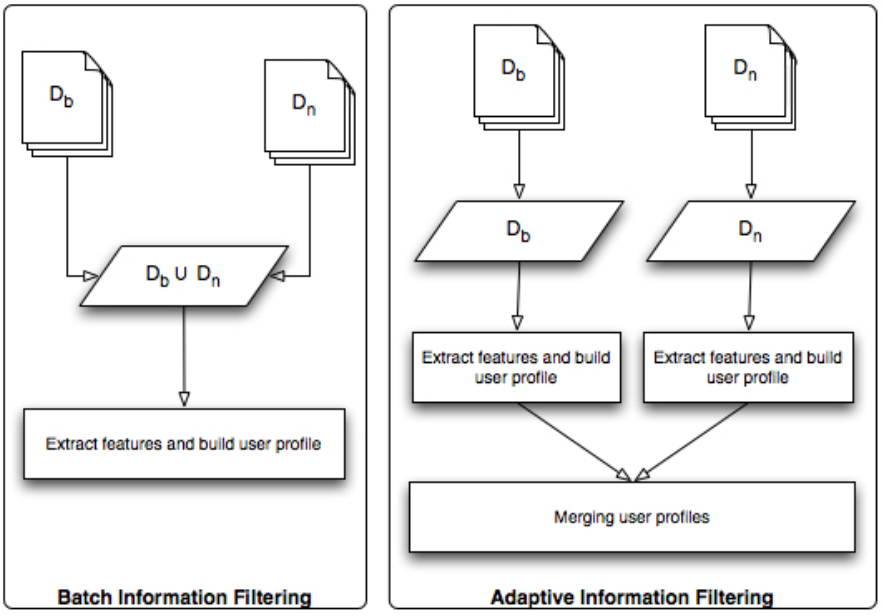

FIGURE 6.10: Adaptive and batch information filtering.

Both the batch and adaptive filtering models use the same training dataset D_b and D_n. However, ARFD uses the two datasets separately utilising adaptive

filtering. On the other hand, other models use training dataset D_b and D_n as one dataset by initially combining both datasets ($D_b \cup D_n$) then employing batch filtering. Both batch and adaptive filtering are illustrated in Figure 6.10. The size of a new training window is set to 25 documents, and for each model, we use six different new training windows to update the system in six loops. In the same loop, all models use the same new training window. The new training window used in one loop will not be chosen for the rest of the loops.

TABLE 6.13: List of methods used for evaluation the ARFD.

Method	Description	Algorithm
ARFD	Adaptive Relevance Feature Discovery Proposed method	SNT-Algorithm Section 5.2
RFD	Relevance Feature Discovery Proposed method	HLFMining, NRevision Section 4.4
Rocchio	Rocchio method	Section 6.3
BM25	Probabilistic method	Section 6.3

6.7.2 Adaptive Result

Table 6.13 shows a list of all methods used for the evaluation of the ARFD model. The results of the proposed model ARFD for all five measures, including time, are presented in Table 6.14. Each row presents the average result for all assessor topics in the RCV1 dataset. In each topic, the system starts from the initial training documents then adds a window of new training documents. The size of the window is set to 25 documents. Each window of training documents is selected randomly. To test the robustness of the proposed model, we also conducted the batch process of RFD for the same initial training set. Table 6.14 shows the

result of the six updates, where the model with "*" is the model that only uses the initial training dataset D_b.

TABLE 6.14: Adaptive Relevance Features Discovery results in all assessor topics.

Model	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*	*Time*
ARFD-1	0.547	0.473	0.492	0.470	0.514	1221.62
ARFD-2	0.534	0.471	0.482	0.464	0.504	1374.36
ARFD-3	0.570	0.489	0.505	0.477	0.524	1376.96
ARFD-4	0.563	0.492	0.508	0.476	0.529	1258.84
ARFD-5	0.547	0.475	0.494	0.470	0.512	1145.38
ARFD-6	0.544	0.478	0.491	0.469	0.511	1184.12
AVG	0.551	0.480	0.495	0.471	0.516	1260.21

TABLE 6.15: Relevance Features Discovery results using batch training documents in all assessor topics.

Model	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*	*Time*
RFD-1	0.585	0.495	0.513	0.483	0.532	7860.94
RFD-2	0.565	0.491	0.512	0.485	0.529	7899.16
RFD-3	0.581	0.486	0.507	0.479	0.528	7820.60
RFD-4	0.575	0.499	0.518	0.484	0.540	7882.86
RFD-5	0.558	0.476	0.497	0.470	0.518	7807.44
RFD-6	0.547	0.475	0.498	0.473	0.519	7817.18
AVG	0.569	0.487	0.508	0.479	0.528	7848.03
RFD*	0.557	0.4724	0.493	0.4696	0.5125	-

Table 6.15 shows the results of the main baseline model (RFD) that is used to compare with the ARFD model. Each loop represents the average result for all assessor topics in the RCV1 dataset. For each topic, the initial training dataset is combined with the same new training dataset that is used in the same topic and the same loop in the ARFD system. The combination of the two datasets is used to train the RFD system from the beginning. Each topic in each loop is run separately. The average time for each run is also calculated and presented

in Table 6.15. The t-test *P-values* of comparison between the two models (RFD and ARFD) are shown in Table 6.16.

TABLE 6.16: t-test *P-value* results comparing batch RFD (Table 6.15) and the ARFD model (Table 6.14) in all assessor topics.

Loop	*Time*	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
Loop-1	5.76039E-07	0.031	0.068	0.182	0.184	0.197
Loop-2	5.78593E-07	0.059	0.087	0.008	0.008	0.017
Loop-3	1.51486E-06	0.467	0.909	0.732	0.721	0.594
Loop-4	1.61788E-06	0.326	0.313	0.168	0.044	0.109
Loop-5	5.68124E-07	0.373	0.993	0.606	0.911	0.392
Loop-6	7.97786E-07	0.733	0.628	0.373	0.332	0.278
AVG	9.42215E-07	0.331	0.500	0.345	0.367	0.264

More results for the state-of-the-art term-based models are shown in Table 6.18 and Table 6.17. The method used to combine the initial training dataset with a new window of training documents is the same as for the batch RFD model. Each topic in each loop for each model uses the same training documents. As mentioned before, the testing is done in six loops. Each loop uses the same initial training dataset with a different new window of the training dataset. The number following the model name in the results table shows the loop number. The model with "*" is the model used only the initial training dataset D_b. Each loop in each model uses the same training dataset.

Table 6.19 and Table 6.20 show the results of the ARFD model and the batch RFD model for the second 50 topics (topic 151 to topic 200). Comparing results in the two tables shows that the proposed model is not a significant improvement on the RFD models in the second 50 topics (topic 151 to topic 200). The most likely reason is that the benchmark judgement of those topics are not accurate. As mentioned previously, the first 50 topics (topic 100 to topic 150) in the RCV1 dataset were judged by human beings and the second 50 topics were judged by

TABLE 6.17: Rocchio model using batch training documents in 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
Rocchio*	0.474	0.4201	0.4305	0.4299	0.4523

TABLE 6.18: BM25 model using batch training documents in all assessor topics.

Model	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*
BM25-1	0.471	0.423558	0.428799	0.431313	0.450396
BM25-2	0.460	0.417241	0.419769	0.426554	0.441968
BM25-3	0.458	0.419341	0.423820	0.424068	0.448159
BM25-4	0.462	0.431403	0.430542	0.429874	0.455036
BM25-5	0.458	0.416261	0.427028	0.423858	0.449629
BM25-6	0.456	0.420646	0.425767	0.425969	0.448193
AVG	0.461	0.421408	0.425954	0.426939	0.448897
BM25*	0.445	0.4074	0.4069	0.4140	0.4281

a machine. Therefore, the results from the first group (assessing topics) are more reliable than the result from the second group. The main objective of the ARFD model is to update the system efficiently while maintaining at least the same level of performance as batch filtering. Referring to the results using the assessing topics, the proposed model ARFD achieved the design objectives.

6.7.3 Discussion

The main process of ARFD aims to update and revise feature weights in vector space efficiently. In this section, we discuss the issues of using more training

TABLE 6.19: Relevance Features Discovery model results from topic 151 to topic 200.

Model	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*	*Time*
RFD-1	0.625	0.523	0.559	0.495	0.574	6645.68
RFD-2	0.627	0.517	0.565	0.499	0.579	6513.98
RFD-3	0.629	0.522	0.562	0.498	0.574	6716.88
RFD-4	0.614	0.515	0.559	0.495	0.574	6666.92
RFD-5	0.631	0.517	0.556	0.494	0.568	6751.88
RFD-6	0.637	0.520	0.565	0.502	0.577	6538.04
AVG	0.627	0.519	0.561	0.497	0.574	6638.90

TABLE 6.20: Adaptive Relevance Features Discovery model results from topic 151 to topic 200.

Model	*top-20*	*b/p*	*MAP*	$F_{\beta=1}$	*IAP*	*Time*
ARFD-1	0.594	0.504	0.540	0.486	0.554	1638.6
ARFD-2	0.591	0.497	0.532	0.482	0.545	1516.82
ARFD-3	0.598	0.504	0.541	0.487	0.553	1626.22
ARFD-4	0.596	0.506	0.536	0.483	0.549	1538.48
ARFD-5	0.589	0.500	0.532	0.480	0.546	1587.5
ARFD-6	0.608	0.514	0.548	0.492	0.563	1613.8
AVG	0.596	0.504	0.538	0.485	0.552	1586.90

documents and the use of the proposed model to speed up the updating time.

6.7.3.1 Adaptive Versus Constant

Generally, using more training documents will lead to an improvement in system effectiveness. This has been supported by the results of experiments conducted using the proposed and baseline models as shown in Table 6.15, 6.14, 6.17 and 6.18. However, the percentage of improvement is affected by several factors.

From results provided in Table 6.15, we can compare using only the initial training dataset with using both the initial dataset plus 25 additional new documents. It is clear that the pattern-based model RFD that uses 25 new documents shows

improvements from $1.01\% = \frac{0.498-0.493}{0.493}$ to $5.07\% = \frac{0.518-0.493}{0.493}$ in average precision compared to RFD* which used only the initial training dataset. All the six loops use the same number of training documents, however, the percentage of improvement is affected by the quality of the training documents. The quality of the training documents can be measured by how closely those documents represent the user's needs. The number of positive and negative documents that appear in the training dataset also affect system effectiveness.

Comparing also with the traditional term-based model Rocchio and BM25 in Table 6.17 and Table 6.18. The table shows that using more 25 training documents can improve the overall system effectiveness by about $4.68\% = \frac{0.455004-0.4305}{0.4305}$ and $5.69\% = \frac{0.425954-0.4069}{0.4069}$ in average precision respectively.

From the above analysis, it seems that the percentage of improvement in term-based approaches achieved by using more training documents is much greater than that of the pattern-base models. However, it is difficult to say that, term-based approaches are more reliable to use in an adaptive system than a pattern-based approach because the pattern-based model (RFD) initially gave a better result. More research needs to be done to investigate this issue.

6.7.3.2 Document Selection

One objective of updating user profiles using new feedback documents in the ARFD model is to solve the nonmonotonic problem. For example, given the system's limited number of training documents about the"Agent", the IF systems may return information objects such as "Intelligent Agent", "Property Agent", or "Software Agent". However, when more information about the user's actual

information needs is obtained later on, the system can determine that the user is only interested in "Software Agent" [32]. Therefore, some of the new training documents will be selected using the knowledge currently held by the system. As shown in Table 6.21 the average number of selected documents in all assessing topics is about 26.2% = $\frac{6.55}{25}$ out of 25 documents provided. It is clear that the size of the window is changeable for each topic. Then different topic would have different window size after selecting those documents. It was shown that about 74% of documents contain the same knowledge that D_b has. It will cost more time to extract knowledge and update user profiles, if the system uses all new feedback documents.

TABLE 6.21: Statistical information about document selection in ARFD in all assessing topics.

Model	$\lvert D_s \rvert$	# Topic used D_s	# Topic did not use D_s
ARFD-1	6.22	15	35
ARFD-2	6.88	18	32
ARFD-3	7.1	18	32
ARFD-4	6.56	17	33
ARFD-5	6.2	16	34
ARFD-6	6.36	19	31
AVG	6.55	17.17	32.83

In addition, Table 6.21 shows that not all extracted knowledge is used to update the system. Instead only about 17 topics out of 50 topics used the extracted knowledge to update the system knowledge obtained from D_b. To determine whether or not to use the new knowledge to update the system, the system updates the existing knowledge with the new knowledge then uses the new training documents D_n to test the system. If the system gets better result compared to using only D_b, then the new knowledge will be used; otherwise, the new knowledge will be ignored.

Based on the above analysis and Table 6.21, not all provided feedback documents are suitable to update the system. Also, not all new knowledge feeds to the system is help to improve the effectiveness of the learning model.

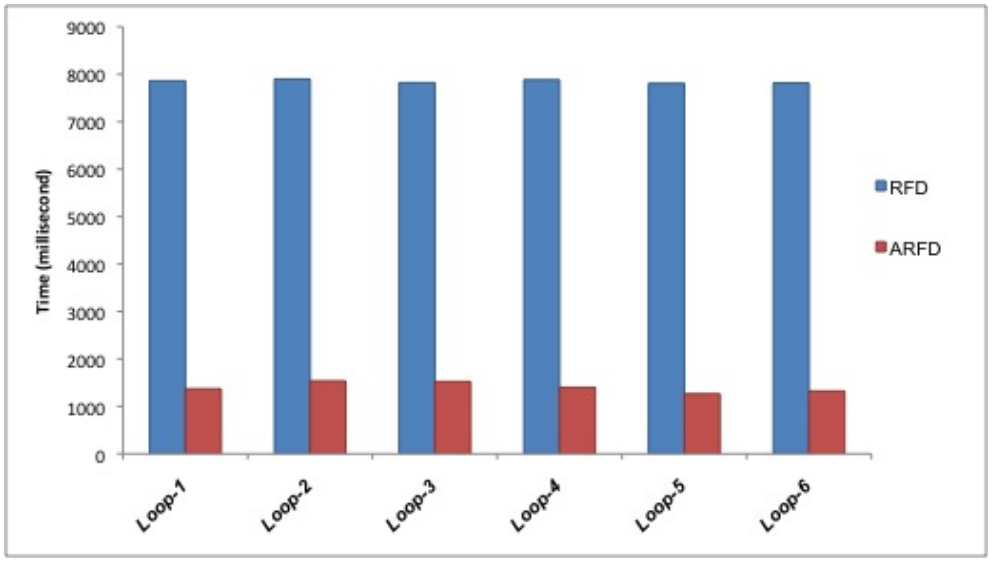

FIGURE 6.11: Time comparison result between RFD and ARFD.

6.7.3.3 Efficiency Versus Effectiveness

Generally, results presented in Table 6.15 and Table 6.14 show that using more training documents would lead to improvements in the effectiveness of the system. The percentage of improvement is affected by the quality of the training documents. In order to see the improvement achieved by the proposed model, we compare it with the RFD model using new feedback documents and batch filtering. Both models use the same training dataset but different methods. As shown in Table 6.15 and Table 6.14, using batch training documents would lead to greater improvement than our adaptive filtering ARFD. However, Table 6.16 clarifies that the improvement is not significant and it does not deserve the time

taken. Comparing the RFD result with the ARFD result, the average of the t-test *p-value* results in all five measures is greater than 0.05 as shown in Table 6.16. It shows that the differences between RFD that uses batch training documents and ARFD that uses the proposed model are not significant.

Our approach aims to update the system with a new training dataset efficiently to improve the effectiveness and solve the nonmonotonic problem. As we saw previously, the effectiveness is almost the same for RFD and ARFD. To measure the efficiency, we compare the time it takes to update the system using RFD and ARFD. The times taken for each loop are presented in Table 6.15 and Table 6.14. On average, RFD takes about $72.33\% = \frac{7848.03-1260.21}{1260.21+7848.03}$ more time than ARFD. The t-test *p-values* in Table 6.16 show that the differences between times taken are strongly significant. Comparison result between times in each loop for RFD and ARFD are presented in Figure 6.11. The huge differences in times between RFD and ARFD can be seen clearly in the figure.

It is true that term based models are more efficient than data mining models. The challenging issue is how to improve the effectiveness of data mining models efficiently. For the efficiency, the proposed model is more expensive in the training phase; but has the same efficiency of IR models in the testing phase. As a result, we did not compare BM25 and Rocchio with the proposed model because BM25 and Rocchio are term-based approaches, therefore it is not fair to compare their times against times taken by the more complex pattern-based model. Finally, the result shows that the proposed model achieves the design objectives. It can reduce the updating process time and maintain almost the same performance for filtering tasks.

6.7.4 Summary

An adaptive information filtering system called adaptive relevance features discovery has been proposed. The ARFD model is built upon the RFD method. The main aim of this method is the efficient revision and updating weight of extracted features in vector space; using new training documents to solve the nonmonotonic problem. A method of selecting useful training documents from the new training dataset was developed. From these selected training documents, new knowledge is extracted. Different methods have been used to merge the new knowledge with the base knowledge. The combination of the old and new knowledge is then tested to ensure that it helps to solve the nonmonotonic problem. Compared with the baseline models that use batch training documents, the experiments on RCV1 and TREC topics demonstrate that the efficiency of updating the system using the proposed model is a significant improvement whilst maintaining almost the same level of effectiveness. Experiments also show that the proposed approach can work efficiently to achieve encouraging performance times for system updates.

Chapter 7

Conclusion

The major research issue in this thesis is how to use negative feedback to improve the quality of extracted features from positive feedback documents. The negative feedback is used to significantly reduce the impact of noisy features among the features extracted from the positive feedback.

Several attempts have used negative feedback to solve this challenge; however, there are two issues associated with using negative relevance feedback to improve the effectiveness of information filtering. The first is how to select constructive negative samples in order to reduce the space of the negative documents. The second issue is how to determine which noisy extracted features should be updated based on the selected negative samples. This thesis presents the research on the concept of developing an effective pattern-based Relevance Feature Discovery model (RFD) that uses both positive and negative feedback. We also extend the RFD to work as an adaptive model.

The RFD model started by selecting some offenders from the negative documents, where an offender can be used to reduce the side affects of noisy features. It also classifies extracted features (e.g. low-level terms) into three categories: positive specific terms, general terms, and negative specific terms. In this way, multiple revising tactics can be used to update 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 with 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.

In the ARFD model, some of the new training documents will be selected using the knowledge currently held by the system. Then, specific features will be extracted from selected training documents. Different methods have been used to merge and revise the weights of features in a vector space. The combination of the old and new knowledge will be tested to ensure that it helps to solve the non-monotonic problem. Compared with the baseline models that use batch training documents, the experiments on RCV1 and TREC topics demonstrate that the efficiency of updating the system using the proposed model is significantly improved and maintains almost the same level of effectiveness. Experiments also show that the proposed approach can work efficiently to achieve encouraging performance times for system updates.

Section 7.1 presents the main contributions of this research. Section 7.2 discusses the possible directions for the future work in the area of this research.

7.1 Contribution

There are two models proposed in this thesis: Relevance Feature Discovery (RFD) and Adaptive Relevance Feature Discovery (ARFD). The contributions are listed as follows:

Relevance Feature Discovery

- **An effective Relevance Feature Discovery model (RFD) is proposed.**

  This research proposes a pattern mining based approach that uses positive feedback documents and selects some offenders from the negative documents, where an offender can be used to select constructive negative samples in order to reduce the space of negative documents. Also, offenders can be used to reduce the side affects of noisy features by classifying extracted features into three categories. In this way, multiple revising tactics can be used to update extracted features.

- **Introduced a strategy to use negative feedback documents.**

  One main problem with using negative feedback is that negative feedback has no clear boundary and the diversity of negative feedback is very high. This research uses the concept of offenders. Offenders can be defined as negative documents that are very close to the positive feedback. The use of offenders can effectively reduce the diversity of negative documents in order to accurately revise feature weights.

- **A new strategy to evaluate the specificity of extracted features.**

  To measure the relevance of specific terms to what users need, this research introduced a specificity function. The specificity function measures the distance between the terms and what the user needs. It is calculated based on its appearance in the training set.

- **Proposed a new strategy to isolate different kinds of knowledge extracted from relevance feedback.**

  Features extracted from positive documents and offenders consist of three kinds of knowledge: positive, negative and general. Positive knowledge is the knowledge that describes what users exactly need. General knowledge includes features that are required to describe what users need like background knowledge, but, these features may also appear in negative documents. Negative knowledge consists of features extracted from offenders that are not what users need. To isolate each kind of knowledge, this research introduces a classification rule that group extracted features into three groups based on specificity score. The three groups are: specific positive, specific negative and general groups.

- **Provided a promising methodology for evaluating term weights based on discovered patterns and specificity score.**

  To more accurately weight the low-level features in the RFD model, an algorithm to revise low-level term weights has been proposed in this research. It is clear that features appearing in the positive group are the most important features. On the other hand, features appearing in the negative group are the less important features. Based on that, the weights of features in

the positive group are increased and decreased for those in the negative group.

- **A Feasible Information Filtering System.**

  An information filtering framework based on the proposed Relevance Feature Discovery model is established and evaluated by a series of experiments. By comparing it to traditional information filtering methods, the RFD model can improve the effectiveness of the system. It also gains the advantages over the up-to-date data mining-based methods such as sequential phrases and frequent itemsets-based methods and the PTM. The experimental results also verify that the proposed system is promising for the challenging issue for the text mining community, that is, to provide effective methods to overcome the limitations of term-based information filtering models and make use of negative relevance feedback. Furthermore, the experiments are conducted on all the topics in the RCV1 dataset, which is the latest benchmark data collection in the area of text mining [70].

Adaptive Relevance Feature Discovery

- **Adaptive Relevance Feature Discovery model (ARFD) is proposed.**

  The research proposes a new adaptive model based on the relevance feature discovery model. It automatically updates the system's knowledge, based on a sliding window over positive and negative feedback, to solve a nonmonotonic problem efficiently.

- **A method to evaluate new feedback has been proposed.**

  It is clear that not all new feedback is useful to update the system knowledge because some of the new training documents have the same knowledge that system obtained from previous training documents. Therefore, this research introduces a new strategy to evaluate the new feedback. The proposed evaluation method ranks the new feedback using the knowledge that system already has. The new documents that can be classified correctly are become useless because the system has knowledge about them.

- **Update the learning model knowledge efficiently.**

  Generally, features extracted from the new feedback have some overlaps with existing knowledge held by the system. How to combine these two features are a challenging task. This research introduces a new strategy to combine these features. It also uses an evaluation method to measure the effectiveness of the combined knowledge.

- **Experimental evaluation is made and the results prove the feasibility and effectiveness of the proposed ARFD model.**

  A new adaptive information filtering framework is proposed. The new model is designed for Relevance Features Discovery (RFD), a pattern mining based approach, which uses negative relevance feedback to improve the quality of extracted features from positive feedback. Compared with the baseline models that use batch training documents, the experiments on RCV1 and TREC topics demonstrate that the efficiency of updating the system using the proposed model is significantly improved whilst maintaining almost the same level of effectiveness. Experiments show that the

proposed approach can work efficiently to achieve encouraging performance times for system updates.

7.2 Future Work

It has been proven that a text document includes a lot of information regarding to different kinds of topics (or knowledge). However, not all information in the document is useful for learning relevant features to a given topic. To improve the quality of extracted features, the proposed model used patterns to select term features because patterns have the good statistical properties. Our experiments showed that selected patterns are good for representing documents but not good enough to represent queries for answering what users want.

We also found that it is easy by using negative feedback to select specific term features for describing what user need. The proposed model proved that long patterns do contain more positive specific features. As mentioned previously, each document includes different kind of information (or knowledge), such as background (general) knowledge and/or specific knowledge. Both the background knowledge and the specific knowledge are necessary for solving the problem of information filtering and information retrieval. The proposed RFD model introduced a new method to define different kinds of knowledge in a set of documents, i.e., S^+, G and S^-. However, both positive and negative feedback are not always available in some real applications. Therefore, it is desire to have a method to isolate different kinds of knowledge by using a single class (e.g., positive only) of documents.

Moreover, the RFD model used two empirical parameters to perform its tasks. Future research work related to this study could be a good way to decide the two parameters in order to accurately group extracted features into different kind of categories.

The framework and analysis study provided in this thesis open a direction for finding different kinds of knowledge in a set of documents in order to improve the quality of feature selection. In the future, we can consider more semantic information and provide more kinds of knowledge for a certain of application.

Appendix A

Details Result

TABLE A.1: Details of results for each topic in the 50 assessing topics in the RCV1 dataset for the RFD model.

Topic	Top_{20}	p/b	MAP	F_{score}	IAP	Recall
r101	1.000	0.79153	0.88254	0.63967	0.87381	0.50163
r102	1.000	0.79874	0.85462	0.63339	0.84433	0.50314
r103	0.900	0.73770	0.82219	0.62814	0.79727	0.50820
r104	0.950	0.65957	0.74011	0.60058	0.73682	0.50532
r105	0.850	0.60000	0.72989	0.60045	0.73476	0.51000
r106	0.150	0.09677	0.12969	0.20730	0.15169	0.51613
r107	0.600	0.35135	0.30209	0.38040	0.33415	0.51351
r108	0.450	0.46667	0.42407	0.47247	0.45318	0.53333
r109	0.500	0.32432	0.42492	0.46224	0.46109	0.50676
r110	0.600	0.48387	0.47772	0.49618	0.49501	0.51613
r111	0.350	0.40000	0.29043	0.37607	0.31995	0.53333
r112	0.500	0.50000	0.36775	0.43253	0.40507	0.52500
r113	0.150	0.44286	0.30905	0.38406	0.34917	0.50714
r114	0.700	0.40323	0.42849	0.46489	0.45592	0.50806
r115	0.550	0.38095	0.39720	0.44579	0.41144	0.50794
r116	0.850	0.66667	0.74364	0.60205	0.75571	0.50575
r117	0.850	0.65625	0.72822	0.60375	0.71088	0.51563
r118	0.100	0.07143	0.12268	0.19964	0.14817	0.53571
r119	0.600	0.50000	0.56347	0.53678	0.58335	0.51250
r120	0.900	0.63291	0.68666	0.58076	0.69836	0.50316
r121	0.800	0.65476	0.70209	0.58810	0.70477	0.50595
r122	0.850	0.76471	0.69947	0.58976	0.74245	0.50980
r123	0.400	0.41176	0.40116	0.45645	0.43260	0.52941
r124	0.150	0.15152	0.14705	0.22879	0.16811	0.51515
r125	0.750	0.49242	0.55964	0.53025	0.57181	0.50379
r126	0.900	0.90116	0.92358	0.65121	0.92622	0.50291
r127	0.400	0.38095	0.38528	0.43966	0.41028	0.51190
r128	0.300	0.33333	0.33351	0.40490	0.35991	0.51515
r129	0.700	0.49123	0.49307	0.50080	0.50993	0.50877
r130	0.350	0.43750	0.42351	0.47130	0.44837	0.53125
r131	0.850	0.67568	0.79425	0.61874	0.78925	0.50676
r132	0.150	0.13636	0.09315	0.15813	0.10795	0.52273
r133	0.600	0.53571	0.54669	0.53188	0.56687	0.51786
r134	0.250	0.29851	0.27722	0.35856	0.32649	0.50746
r135	1.000	0.88427	0.92208	0.64965	0.92178	0.50148
r136	0.300	0.22388	0.25118	0.33604	0.32139	0.50746
r137	0.400	0.55556	0.68972	0.61541	0.68226	0.55556
r138	0.400	0.31818	0.30206	0.37978	0.31798	0.51136
r139	0.550	0.52941	0.60112	0.56299	0.60499	0.52941
r140	0.850	0.46269	0.49705	0.50220	0.50447	0.50746
r141	0.400	0.57317	0.53756	0.52135	0.57873	0.50610
r142	0.400	0.33333	0.32959	0.40371	0.36093	0.52083
r143	0.100	0.08696	0.10901	0.18035	0.13455	0.52174
r144	0.800	0.63636	0.69969	0.58936	0.69912	0.50909
r145	0.150	0.18519	0.13864	0.21879	0.19634	0.51852
r146	0.650	0.53153	0.56737	0.53409	0.58708	0.50450
r147	0.550	0.44118	0.49071	0.50242	0.52814	0.51471
r148	1.000	0.88158	0.92857	0.65185	0.92000	0.50219
r149	0.100	0.07018	0.14817	0.22950	0.23068	0.50877
r150	0.200	0.27778	0.24001	0.32625	0.25314	0.50926

TABLE A.3: Details of results of 11-points for each topic in the all assessing topics in the RCV1 dataset for the RFD model.

Topic	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
r101	1	0.9756098	0.97183096	0.9489796	0.93333334	0.92771083	0.87096775	0.83657587	0.7942122	0.7679558	0.5847619
r102	1	1	0.9142857	0.89855075	0.87777776	0.86538464	0.84210527	0.8169014	0.7950311	0.742268	0.53535354
r103	1	1	0.9444444	0.9310345	0.9310345	0.8717949	0.8636364	0.7962963	0.6329114	0.6111111	0.18769231
r104	1	1	0.95454544	0.8974359	0.82608694	0.80597013	0.7808219	0.6	0.46341464	0.41095892	0.36575875
r105	1	1	1	0.85	0.7777778	0.7222222	0.63829786	0.5633803	0.5633803	0.5	0.46728972
r106	0.1875	0.1513158	0.1513158	0.1513158	0.1513158	0.1513158	0.1513158	0.1513158	0.14583333	0.14583333	0.13025211
r107	1	0.6	0.6	0.6	0.35714287	0.15833333	0.091603056	0.073446326	0.065026365	0.065026365	0.065026365
r108	0.6	0.6	0.6	0.6	0.6	0.5294118	0.5294118	0.42307693	0.2857143	0.10869565	0.10869565
r109	1	0.6	0.3986014	0.3986014	0.3986014	0.3986014	0.3986014	0.3986014	0.38787878	0.36216217	0.33035713
r110	1	1	0.75	0.6666667	0.59090906	0.5	0.31746033	0.2682927	0.16352202	0.09602649	0.0922619
r111	1	0.6666667	0.4375	0.4375	0.4375	0.1875	0.1875	0.046428572	0.046428572	0.036231883	0.036231883
r112	0.5263158	0.5263158	0.5263158	0.5263158	0.5263158	0.5263158	0.29090908	0.29090908	0.29090908	0.28125	0.1438849
r113	0.45070422	0.45070422	0.45070422	0.45070422	0.45070422	0.42528737	0.34351146	0.273743	0.22619048	0.18421052	0.134357
r114	0.7368421	0.7368421	0.7368421	0.67741936	0.4385965	0.3974359	0.30645162	0.2804878	0.24017467	0.23529412	0.22878228
r115	1	0.72727275	0.43396226	0.43396226	0.35365853	0.33663365	0.30952382	0.2760736	0.24285714	0.21189591	0.2
r116	1	0.875	0.85714287	0.8484849	0.78571427	0.78571427	0.67948717	0.67	0.6603774	0.62992126	0.52095807
r117	1	1	1	1	1	0.8888889	0.7777778	0.42105263	0.3561644	0.2248062	0.1509434
r118	0.2	0.2	0.2	0.2	0.2	0.13559322	0.12087912	0.12087912	0.088435374	0.088435374	0.075675674
r119	1	1	0.8	0.61904764	0.52380955	0.52380955	0.47368422	0.46666667	0.4047619	0.3272727	0.2777778
r120	0.9	0.9	0.8979592	0.8909091	0.79012346	0.7017544	0.6535948	0.59358287	0.51626015	0.44859812	0.38916257
r121	1	1	0.8095238	0.7916667	0.7916667	0.7586207	0.6933333	0.6451613	0.6086956	0.42857143	0.22520107
r122	0.8695652	0.8695652	0.8695652	0.8695652	0.8604651	0.8604651	0.8604651	0.8604651	0.68333334	0.30463576	0.25888324
r123	1	1	0.75	0.75	0.44444445	0.23255815	0.16666667	0.15189873	0.097402595	0.08695652	0.0787037
r124	0.33333334	0.16666667	0.15048544	0.15048544	0.15048544	0.15048544	0.15048544	0.15048544	0.15048544	0.15048544	0.14537445
r125	1	0.83870965	0.8	0.63013697	0.62068963	0.49264705	0.42857143	0.41333333	0.38434163	0.3821656	0.2993197
r126	1	0.9577465	0.9577465	0.9577465	0.94871795	0.9411765	0.9385965	0.92805755	0.90849674	0.90229887	0.7478261
r127	0.8	0.71428573	0.48	0.4375	0.4107143	0.4107143	0.4	0.2777778	0.221519	0.18025751	0.18025751
r128	1	0.3529412	0.3529412	0.3529412	0.35	0.30985916	0.30985916	0.28915662	0.25	0.21582733	0.17553191
r129	1	0.9	0.8	0.6785714	0.53488374	0.42857143	0.37634408	0.25	0.25	0.23423423	0.15659341
r130	1	0.75	0.6666667	0.54545456	0.53846157	0.36363637	0.23333333	0.23333333	0.23333333	0.18390805	0.18390805
r131	1	1	0.9411765	0.9032258	0.8979592	0.8979592	0.88235295	0.62650603	0.57391304	0.5564516	0.4021739
r132	0.33333334	0.1764706	0.13636364	0.1	0.06440678	0.06440678	0.06440678	0.06440678	0.06440678	0.06116208	0.058047492
r133	1	1	1	0.75	0.7058824	0.6	0.4047619	0.3125	0.21296297	0.14054054	0.108949415
r134	1	0.3272727	0.3272727	0.30666667	0.3030303	0.28368795	0.27333334	0.19252874	0.19252874	0.19252874	0.19252874
r135	1	0.97619045	0.9464286	0.9464286	0.9415584	0.9263566	0.9263566	0.9263566	0.9127517	0.88150287	0.7556054
r136	1	0.375	0.28	0.24747474	0.24747474	0.24747474	0.24747474	0.24747474	0.2364341	0.2364341	0.17005076
r137	1	1	1	1	1	0.625	0.46153846	0.44444445	0.44444445	0.2647059	0.2647059
r138	0.5	0.5	0.4	0.36206895	0.36206895	0.3188406	0.26605505	0.24060151	0.21556886	0.17982456	0.15277778
r139	1	1	0.8333333	0.75	0.7	0.6923077	0.57894737	0.42857143	0.35897437	0.1904762	0.12230216
r140	1	1	0.8888889	0.75	0.5625	0.2881356	0.2611465	0.24352331	0.20220588	0.19018404	0.16262136
r141	1	0.61538464	0.61538464	0.6136364	0.5822785	0.5822785	0.5591398	0.49166667	0.47142857	0.44252872	0.3923445
r142	1	0.71428573	0.71428573	0.53333336	0.15068494	0.15	0.15	0.15	0.14084508	0.13414635	0.13259669
r143	0.5	0.13793103	0.13333334	0.13333334	0.13333334	0.08609272	0.07194245	0.07194245	0.07194245	0.07012195	0.07012195
r144	1	0.9	0.875	0.8125	0.8125	0.7631579	0.65384614	0.52272725	0.52272725	0.4903846	0.33742332
r145	1	0.25	0.1875	0.11111111	0.11111111	0.10687023	0.07931034	0.07931034	0.07931034	0.077586204	0.077586204
r146	1	0.68421054	0.6097561	0.5882353	0.5882353	0.54901963	0.5413534	0.52597404	0.50282484	0.47186148	0.39642859
r147	1	1	0.6666667	0.59090906	0.47368422	0.47368422	0.375	0.32051283	0.31632653	0.31632653	0.27642277
r148	1	0.96666664	0.96666664	0.96511626	0.94392526	0.9375	0.93333334	0.920904	0.9104478	0.87815124	0.6972477
r149	1	0.17297298	0.17297298	0.17297298	0.17297298	0.17297298	0.14693877	0.1369863	0.12958436	0.12958436	0.12954545
r150	0.33333334	0.33333334	0.30555555	0.2857143	0.28037384	0.28037384	0.275	0.22674419	0.16071428	0.15506329	0.14835165

TABLE A.4: Details of results for each topic in the last 50 topics in the RCV1 dataset for the RFD model.

Topic	Top_{20}	p/b	MAP	F_{score}	IAP	Recall
r151	0.300	0.27273	0.21297	0.30264	0.25814	0.52273
r152	0.550	0.53659	0.56907	0.53914	0.56860	0.51220
r153	0.850	0.59459	0.72908	0.60260	0.74272	0.51351
r154	0.750	0.48718	0.58043	0.54453	0.57135	0.51282
r155	0.350	0.30159	0.25130	0.33624	0.30047	0.50794
r156	1.000	0.84722	0.91590	0.65265	0.90930	0.50694
r157	0.200	0.24324	0.22310	0.31106	0.24382	0.51351
r158	0.250	0.28889	0.22642	0.31382	0.25420	0.51111
r159	1.000	0.72165	0.80393	0.62045	0.79182	0.50515
r160	1.000	0.83333	0.89911	0.65023	0.87485	0.50926
r161	0.600	0.51064	0.52046	0.51550	0.52615	0.51064
r162	0.700	0.62963	0.65534	0.57118	0.67056	0.50617
r163	0.850	0.76230	0.81229	0.62212	0.80942	0.50410
r164	1.000	0.79670	0.87169	0.63770	0.85778	0.50275
r165	0.750	0.59615	0.60420	0.55289	0.60156	0.50962
r166	0.150	0.17647	0.27576	0.36263	0.31235	0.52941
r167	0.600	0.45000	0.50331	0.50786	0.51457	0.51250
r168	0.900	0.87732	0.88669	0.64095	0.90500	0.50186
r169	0.250	0.20000	0.26617	0.35079	0.29174	0.51429
r170	0.450	0.46575	0.46007	0.48233	0.48875	0.50685
r171	0.300	0.26471	0.27307	0.35505	0.29302	0.50735
r172	0.150	0.19512	0.17281	0.25843	0.18287	0.51220
r173	0.950	0.83628	0.89735	0.64400	0.90013	0.50221
r174	0.600	0.54878	0.51360	0.50982	0.53353	0.50610
r175	1.000	0.00000	1.00000	0.66809	1.00000	0.50160
r176	0.450	0.48649	0.40882	0.45522	0.43058	0.51351
r177	0.950	0.80328	0.85738	0.63815	0.84292	0.50820
r178	0.400	0.31915	0.28380	0.36483	0.30049	0.51064
r179	0.400	0.40625	0.29399	0.37447	0.32864	0.51563
r180	0.900	0.72222	0.75632	0.60702	0.75119	0.50694
r181	0.300	0.28000	0.15735	0.24159	0.17401	0.52000
r182	0.250	0.25000	0.36859	0.42988	0.41022	0.51563
r183	0.950	0.69784	0.76903	0.60863	0.76249	0.50360
r184	0.100	0.07692	0.09648	0.16364	0.11274	0.53846
r185	1.000	0.81522	0.90367	0.64604	0.89166	0.50272
r186	1.000	0.79545	0.86683	0.63571	0.86528	0.50189
r187	0.700	0.61290	0.68648	0.58924	0.68726	0.51613
r188	0.900	0.83333	0.89084	0.65179	0.87575	0.51389
r189	0.500	0.67105	0.53194	0.51895	0.58273	0.50658
r190	0.900	0.67059	0.71817	0.59362	0.71684	0.50588
r191	0.050	0.05556	0.06201	0.11099	0.06700	0.52778
r192	0.350	0.34483	0.26494	0.35040	0.28290	0.51724
r193	0.200	0.25000	0.14526	0.22814	0.15905	0.53125
r194	0.850	0.73262	0.76267	0.60596	0.77125	0.50267
r195	0.200	0.18919	0.22304	0.31100	0.25967	0.51351
r196	0.500	0.42000	0.36793	0.42747	0.39737	0.51000
r197	0.900	0.74306	0.82059	0.62405	0.82387	0.50347
r198	0.200	0.16667	0.19843	0.28842	0.23980	0.52778
r199	0.600	0.75000	0.70261	0.58717	0.74808	0.50431
r200	0.600	0.51163	0.53710	0.52099	0.56875	0.50581

TABLE A.6: Details of results of 11-points for each topic in the last 50 topics in the RCV1 dataset for the RFD model.

Topic	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
r151	1	0.5555556	0.5555556	0.25925925	0.08130081	0.07971015	0.06726457	0.06315789	0.06315789	0.058495823	0.05612245
r152	1	0.8333333	0.6923077	0.61764705	0.61764705	0.61764705	0.5208333	0.44615385	0.42857143	0.3490566	0.13141026
r153	1	1	1	0.85714287	0.85	0.7916667	0.6	0.5777778	0.56363636	0.52307695	0.4065934
r154	1	1	1	1	0.6666667	0.46666667	0.37313432	0.31111112	0.20779221	0.16289593	0.096534654
r155	1	0.4117647	0.31666666	0.31666666	0.23076923	0.20231214	0.19387755	0.1882353	0.16558442	0.14352942	0.13577586
r156	1	1	1	0.96666664	0.96666664	0.9318182	0.88	0.85714287	0.85714287	0.85714287	0.6857143
r157	0.6666667	0.25714287	0.25714287	0.2413793	0.22535211	0.21505377	0.18699187	0.16959064	0.16853933	0.16203703	0.13214286
r158	0.34615386	0.34615386	0.34615386	0.34615386	0.34615386	0.24038461	0.23275863	0.20253165	0.17370892	0.13084112	0.08522727
r159	1	1	1	0.9074074	0.9074074	0.9074074	0.86764705	0.74725276	0.59090906	0.44278607	0.33916083
r160	1	1	1	1	1	1	1	1	0.88235295	0.4117647	0.3292683
r161	1	1	0.78571427	0.60714287	0.52272725	0.5217391	0.4225352	0.37362638	0.24836601	0.17269076	0.13314448
r162	1	0.8181818	0.71428573	0.71428573	0.69491524	0.69491524	0.64102566	0.6344086	0.5855856	0.47560975	0.40298507
r163	1	0.9375	0.88	0.88	0.88	0.88	0.84090906	0.8055556	0.7480916	0.625	0.42657343
r164	1	1	0.9767442	0.953125	0.9166667	0.91588783	0.888	0.8716216	0.78918916	0.66938776	0.455
r165	1	0.85714287	0.8	0.8	0.8	0.74285716	0.52459013	0.41111112	0.3255814	0.24352331	0.11231101
r166	1	0.6666667	0.22857143	0.22857143	0.22857143	0.2264151	0.2264151	0.2264151	0.18421052	0.1300813	0.08994709
r167	1	1	1	0.6666667	0.56666666	0.41666666	0.390625	0.23931624	0.16836734	0.114369504	0.09756097
r168	1	0.9217391	0.9217391	0.9217391	0.91525424	0.9026549	0.9026549	0.9026549	0.8972332	0.8781362	0.7911765
r169	1	0.2777778	0.26086956	0.26086956	0.26086956	0.26086956	0.25274727	0.22321428	0.16853933	0.13333334	0.11006289
r170	1	0.6666667	0.60465115	0.60465115	0.54545456	0.4512195	0.4	0.3561644	0.3206522	0.275	0.15176715
r171	0.5	0.37142858	0.35714287	0.2987013	0.2972973	0.2857143	0.27702704	0.24170616	0.22580644	0.18658893	0.18181819
r172	0.2857143	0.20833333	0.20833333	0.2	0.2	0.2	0.16455697	0.16393442	0.15348837	0.13405797	0.09318182
r173	1	0.975	0.975	0.975	0.92	0.90625	0.89171976	0.8804348	0.8551402	0.8031496	0.7197452
r174	0.78571427	0.78571427	0.6666667	0.5714286	0.5689655	0.5529412	0.52577317	0.4427481	0.3723404	0.37	0.22651933
r175	1	1	1	1	1	1	1	1	1	1	1
r176	0.5714286	0.5714286	0.5625	0.56	0.5483871	0.5116279	0.5	0.34567901	0.2970297	0.13565892	0.13261649
r177	1	1	1	0.96666664	0.96666664	0.9591837	0.9591837	0.9591837	0.8448276	0.3416149	0.27477476
r178	0.5	0.46153846	0.3611111	0.3409091	0.2638889	0.24242425	0.24242425	0.24242425	0.24242425	0.22395833	0.18431373
r179	0.625	0.625	0.5	0.41935483	0.41935483	0.30357143	0.25974026	0.15231788	0.124423966	0.11788618	0.06837607
r180	1	0.9166667	0.9166667	0.9166667	0.85294116	0.84090906	0.8148148	0.75	0.63736266	0.4370861	0.18
r181	0.33333334	0.33333334	0.31578946	0.21428572	0.2037037	0.16867469	0.098684214	0.077586204	0.07194245	0.052154195	0.04464286
r182	1	0.5714286	0.33333334	0.33333334	0.33333334	0.33333334	0.32857144	0.32857144	0.31683168	0.31683168	0.31683168
r183	1	0.95238096	0.9354839	0.8627451	0.8023256	0.8021978	0.74336284	0.6993007	0.6725146	0.5550661	0.36197916
r184	0.14285715	0.14285715	0.14285715	0.14285715	0.14285715	0.14285715	0.14285715	0.067073174	0.067073174	0.058252428	0.04779412
r185	1	1	1	1	0.9883721	0.96907216	0.94067794	0.88	0.8277778	0.6720648	0.5302594
r186	1	1	0.9661017	0.95348835	0.90082645	0.8815789	0.84771574	0.82300884	0.7888889	0.71257484	0.6439024
r187	1	1	0.9166667	0.9166667	0.7647059	0.61538464	0.61290324	0.5106383	0.50980395	0.47457626	0.23846154
r188	1	1	1	1	1	0.9375	0.9375	0.9375	0.9375	0.76744187	0.115755625
r189	0.67948717	0.67948717	0.67948717	0.67948717	0.67948717	0.67948717	0.67948717	0.675	0.49593496	0.26492536	0.21776505
r190	1	0.93333334	0.9	0.8125	0.75	0.7288136	0.69512194	0.6666667	0.6448598	0.47530866	0.27868852
r191	0.078431375	0.078431375	0.078431375	0.078431375	0.078431375	0.06666667	0.056140352	0.056140352	0.056140352	0.05487805	0.05487805
r192	0.44444445	0.44444445	0.4117647	0.3548387	0.35135135	0.3125	0.2247191	0.20192307	0.16783217	0.1125	0.085545726
r193	0.26666668	0.26666668	0.26666668	0.23809524	0.17073171	0.14814815	0.09322034	0.08974359	0.08974359	0.073529415	0.046376813
r194	1	0.87096775	0.8666667	0.86238533	0.86238533	0.86238533	0.8308824	0.74857146	0.6880734	0.51367784	0.37777779
r195	1	0.26923078	0.18343195	0.18343195	0.18343195	0.18343195	0.18343195	0.18343195	0.18343195	0.15450644	0.14859438
r196	0.6	0.6	0.56	0.51724136	0.5121951	0.4385965	0.28947368	0.27131784	0.24852072	0.20454545	0.12919897
r197	1	0.9375	0.9117647	0.8518519	0.84761906	0.84761906	0.84761906	0.816	0.73417723	0.6806283	0.5877551
r198	0.33333334	0.33333334	0.33333334	0.33333334	0.33333334	0.33333334	0.23913044	0.12871288	0.09497207	0.09497207	0.08
r199	1	0.76033056	0.76033056	0.76033056	0.76033056	0.76033056	0.76033056	0.76033056	0.7580645	0.6730769	0.47540984
r200	1	0.7096774	0.7096774	0.65	0.51960784	0.51960784	0.51960784	0.46616542	0.43125	0.40609136	0.3245283

Appendix B

Topic Codes of TREC RCV1 dataset

CODE	DESCRIPTION
1POL	CURRENT NEWS - POLITICS
2ECO	CURRENT NEWS - ECONOMICS
3SPO	CURRENT NEWS - SPORT
4GEN	CURRENT NEWS - GENERAL
6INS	CURRENT NEWS - INSURANCE
7RSK	CURRENT NEWS - RISK NEWS
8YDB	TEMPORARY
9BNX	TEMPORARY
ADS10	CURRENT NEWS - ADVERTISING
BNW14	CURRENT NEWS - BUSINESS NEWS
BRP11	CURRENT NEWS - BRANDS
C11	STRATEGY/PLANS

C12	LEGAL/JUDICIAL
C13	REGULATION/POLICY
C14	SHARE LISTINGS
C15	PERFORMANCE
C151	ACCOUNTS/EARNINGS
C1511	ANNUAL RESULTS
C152	COMMENT/FORECASTS
C16	INSOLVENCY/LIQUIDITY
C17	FUNDING/CAPITAL
C171	SHARE CAPITAL
C172	BONDS/DEBT ISSUES
C173	LOANS/CREDITS
C174	CREDIT RATINGS
C18	OWNERSHIP CHANGES
C181	MERGERS/ACQUISITIONS
C182	ASSET TRANSFERS
C183	PRIVATISATIONS
C21	PRODUCTION/SERVICES
C22	NEW PRODUCTS/SERVICES
C23	RESEARCH/DEVELOPMENT
C24	CAPACITY/FACILITIES
C31	MARKETS/MARKETING
C311	DOMESTIC MARKETS
C312	EXTERNAL MARKETS
C313	MARKET SHARE

C32	ADVERTISING/PROMOTION
C33	CONTRACTS/ORDERS
C331	DEFENCE CONTRACTS
C34	MONOPOLIES/COMPETITION
C41	MANAGEMENT
C411	MANAGEMENT MOVES
C42	LABOUR
CCAT	CORPORATE/INDUSTRIAL
E11	ECONOMIC PERFORMANCE
E12	MONETARY/ECONOMIC
E121	MONEY SUPPLY
E13	INFLATION/PRICES
E131	CONSUMER PRICES
E132	WHOLESALE PRICES
E14	CONSUMER FINANCE
E141	PERSONAL INCOME
E142	CONSUMER CREDIT
E143	RETAIL SALES
E21	GOVERNMENT FINANCE
E211	EXPENDITURE/REVENUE
E212	GOVERNMENT BORROWING
E31	OUTPUT/CAPACITY
E311	INDUSTRIAL PRODUCTION
E312	CAPACITY UTILIZATION
E313	INVENTORIES

E41	EMPLOYMENT/LABOUR
E411	UNEMPLOYMENT
E51	TRADE/RESERVES
E511	BALANCE OF PAYMENTS
E512	MERCHANDISE TRADE
E513	RESERVES
E61	HOUSING STARTS
E71	LEADING INDICATORS
ECAT	ECONOMICS
ENT12	CURRENT NEWS - ENTERTAINMENT
G11	SOCIAL AFFAIRS
G111	HEALTH/SAFETY
G112	SOCIAL SECURITY
G113	EDUCATION/RESEARCH
G12	INTERNAL POLITICS
G13	INTERNATIONAL RELATIONS
G131	DEFENCE
G14	ENVIRONMENT
G15	EUROPEAN COMMUNITY
G151	EC INTERNAL MARKET
G152	EC CORPORATE POLICY
G153	EC AGRICULTURE POLICY
G154	EC MONETARY/ECONOMIC
G155	EC INSTITUTIONS
G156	EC ENVIRONMENT ISSUES

G157	EC COMPETITION/SUBSIDY
G158	EC EXTERNAL RELATIONS
G159	EC GENERAL
GCAT	GOVERNMENT/SOCIAL
GCRIM	CRIME, LAW ENFORCEMENT
GDEF	DEFENCE
GDIP	INTERNATIONAL RELATIONS
GDIS	DISASTERS AND ACCIDENTS
GEDU	EDUCATION
GENT	ARTS, CULTURE, ENTERTAINMENT
GENV	ENVIRONMENT AND NATURAL WORLD
GFAS	FASHION
GHEA	HEALTH
GJOB	LABOUR ISSUES
GMIL	MILLENNIUM ISSUES
GOBIT	OBITUARIES
GODD	HUMAN INTEREST
GPOL	DOMESTIC POLITICS
GPRO	BIOGRAPHIES, PERSONALITIES, PEOPLE GREL RELIGION
GSCI	SCIENCE AND TECHNOLOGY
GSPO	SPORTS
GTOUR	TRAVEL AND TOURISM
GVIO	WAR, CIVIL WAR
GVOTE	ELECTIONS
GWEA	WEATHER

GWELF	WELFARE, SOCIAL SERVICES
M11	EQUITY MARKETS
M12	BOND MARKETS
M13	MONEY MARKETS
M131	INTERBANK MARKETS
M132	FOREX MARKETS
M14	COMMODITY MARKETS
M141	SOFT COMMODITIES
M142	METALS TRADING
M143	ENERGY MARKETS
MCAT	MARKETS
MEUR	EURO CURRENCY
PRB13	CURRENT NEWS - PRESS RELEASE WIRES

Appendix C

List of Stopwords

a, about, above, according, across, after, afterwards, again, against, albeit, all, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anywhere, apart, are, around, as, at, av, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, below, beside, besides, between, beyond, both, but, by, can, cannot, canst, certain, cf, choose, contrariwise, cos, could, cu, day, do, does, doesn, doing, dost, doth, double, down, dual, during, each, either, else, elsewhere, enough, et, etc, even, ever, every, everybody, everyone, everything, everywhere, except, excepted, excepting, exception, exclude, excluding, exclusive, far, farther, farthest, few, ff, first, for, formerly, forth, forward, from, front, further, furthermore, furthest, get, go, had, halves, hardly, has, hast, hath, have, he, hence, henceforth, her, here, hereabouts, hereafter, hereby, herein, hereto, hereupon, hers, herself, him, himself, hindmost, his, hither, hitherto, how, however, howsoever, i, ie, if, in, inasmuch, inc, include, included, including, indeed,

indoors, inside, insomuch, instead, into, inward, inwards, is, it, its, itself, just, kind, kg, km, last, latter, latterly, less, lest, let, like, little, ltd, many, may, maybe, me, meantime, meanwhile, might, moreover, most, mostly, more, mr, mrs, ms, much, must, my, myself, namely, need, neither, never, nevertheless, next, no, nobody, none, nonetheless, noone, nope, nor, not, nothing, notwithstanding, now, nowadays, nowhere, of, off, often, ok, on, once, one, only, onto, or, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, own, per, perhaps, plenty, provide, quite, rather, really, reuter, reuters, round, said, sake, same, sang, save, saw, see, seeing, seem, seemed, seeming, seems, seen, seldom, selves, sent, several, shalt, she, should, shown, sideways, since, slept, slew, slung, slunk, smote, so, some, somebody, somehow, someone, something, sometime, sometimes, somewhat, somewhere, spake, spat, spoke, spoken, sprang, sprung, stave, staves, still, such, supposing, than, that, the, thee, their, them, themselves, then, thence, thenceforth, there, thereabout, thereabouts, thereafter, thereby, therefore, therein, thereof, thereon, thereto, thereupon, these, they, this, those, thou, though, thrice, through, throughout, thru, thus, thy, thyself, till, to, together, too, toward, towards, ugh, unable, under, underneath, unless, unlike, until, up, upon, upward, upwards, us, use, used, using, very, via, vs, want, was, we, week, well, were, what, whatever, whatsoever, when, whence, whenever, whensoever, where, whereabouts, whereafter, whereas, whereat, whereby, wherefore, wherefrom, wherein, whereinto, whereof, whereon, wheresoever, whereto, whereunto, whereupon, wherever, wherewith, whether, whew, which, whichever, whichsoever, while, whilst, whither, who, whoa, whoever, whole, whom, whomever, whomsoever, whose, whosoever, why, will, wilt, with, within, without, worse, worst, would, wow, ye, yet, year, yippee, you, your, yours, yourself, yourselves

Bibliography

[1] K. Aas and L. Eikvil. Text categorisation: A survey., 1999.

[2] X. T. Abdulmohsen Algarni, Yuefeng Li. Mining Specific and General Features in Both Positive and Negative Relevance Feedback. In *Text Retrieval Conference (TREC 2009)*, 2009.

[3] R. Agrawal and G. Shafer. Parallel mining of association rules. *Knowledge and Data Engineering, IEEE Transactions on*, 8(6):962–969, 1996.

[4] H. Ahonen, O. Heinonen, M. Klemettinen, and A. Inkeri Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In *Proceedings of the Advances in Digital Libraries Conference; ADL '98:*, pages 2–11. IEEE Computer Society, 1998.

[5] H. Ahonen, O. Heinonen, M. Klemettinen, and A. I. Verkamo. Mining in the phrasal frontier. *PKDD '97: Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 343–350, 1997.

[6] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation, 2002.

[7] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[8] R. A. Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. 1999.

[9] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Research and development in information retrieval*, SIGIR '94, pages 292–300. Springer-Verlag New York, Inc., 1994.

[10] J. Callan. Learning while filtering documents. In *Research and development in information retrieval; SIGIR '98*, pages 224–231. ACM, 1998.

[11] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Research and development in information retrieval*, SIGIR '08, pages 243–250. ACM, 2008.

[12] L. Carsten. Evaluating performance indicators for adaptive information filtering, 1999.

[13] G. Chen, X. Wu, and X. Zhu. Sequential pattern mining in multiple streams. *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 585–588, 2005.

[14] M. L. Chris Buckley and M. D. Smucker. Overview of the TREC 2010 Relevance Feedback Track. In *Text Retrieval Conference (TREC 2010)*, 2010.

[15] K. W. Church. One term or two? In *Research and development in information retrieval, 1995*, SIGIR '95, pages 310–318. ACM, 1995.

[16] K. J. Cios. *Data mining : a knowledge discovery approach*. Springer, New York, 2007.

[17] A. Dasgupta, P. Drineas, B. Harb, V. Josifovski, and M. W. Mahoney. Feature selection methods for text classification. In *Knowledge discovery and data mining; KDD '07*, pages 230–239. ACM, 2007.

[18] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23:229–236, 1991.

[19] D. A. Evans, J. Shanahan, N. Roma, J. Bennett, V. Sheftel, E. Stoica, J. Montgomery, D. A. Hull, and W. Tembe. Clarit experiments in batch filtering: Term selection and threshold optimization in ir and svm filters. *TREC02*, 2002.

[20] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, 1996.

[21] X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 106–112. ACM, 2000.

[22] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using k-nearest neighbor. In *Research and development in information retrieval; SIGIR '08*, pages 115–122. ACM, 2008.

[23] J. Han and K. Chang. Data mining for web intelligence. *IEEE Computer*, 35(11):64–70, 2002.

[24] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Management of data; SIGMOD '00*, pages 1–12. ACM, 2000.

[25] D. He, P. Brusilovsky, J. Ahn, J. Grady, R. Farzan, Y. Peng, Y. Yang, and M. Rogati. An evaluation of adaptive filtering in the context of realistic task-based information exploration. *Inf. Process. Manage.*, 44:511–533, March 2008.

[26] Y.-F. Huang and S.-Y. Lin. Mining sequential patterns using graph search techniques. *Computer Software and Applications Conference, Annual International*, 0:4–9, 2003.

[27] G. Ifrim, G. Bakir, and G. Weikum. Fast logistic regression for text categorization with variable-length n-grams. In *Knowledge discovery and data mining; KDD '08:*, pages 354–362. ACM, 2008.

[28] N. Jindal and B. Liu. Identifying comparative sentences in text documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 244–251, New York, NY, USA, 2006. ACM.

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

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

[31] C. Lanquillon and I. Renz. Adaptive information filtering: detecting changes in text streams, 1999.

[32] R. Lau, A. H. M. ter Hofstede, and P. D. Bruza. Nonmonotonic reasoning for adaptive information filtering. In *Computer science; ACSC '01*, pages 109–116. IEEE Computer Society, 2001.

[33] R. Y. Lau, P. D. Bruza, and D. Song. Belief revision for adaptive information retrieval. pages 130–137, 2004.

[34] V. Lavrenko and W. B. Croft. Relevance based language models. In *Research and development in information retrieval*, SIGIR '01, pages 120–127. ACM, 2001.

[35] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '92, pages 37–50. ACM, 1992.

[36] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December 2004.

[37] H. Li and w. Yin. Study of application of web mining techniques in e-business. In w. Yin, editor, *Service Systems and Service Management, 2006 International Conference on*, volume 2, pages 1587–1592, 2006.

[38] X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 587–592. Morgan Kaufmann Publishers Inc., 2003.

[39] Y. Li, A. Algarni, and N. Zhong. Mining positive and negative patterns for relevance feature discovery. In *knowledge discovery and data mining*, KDD '10, pages 753–762. ACM, 2010.

[40] Y. Li, X.-Z. Chen, and B.-R. Yang. Research on web mining-based intelligent search engine. volume 1 of *Proceedings of 2002 International Conference on Machine Learning and Cybernetics*, pages 386–390, Beijing, China, 2002. Institute of Electrical and Electronics Engineers Inc.

[41] Y. Li, S. Wu, and Y. Xu. Deploying association rules on hypothesis spaces. In M. Mohammadian, editor, *Computioonal Intelligence for Modelling, Control and Automation (CIMCA 2004)*, pages 769–778, Gold Coast, QLD, 2004. University of Canberra.

[42] Y. Li, C. Zhang, and J. R. Swan. An information filtering model on the web and its application in jobagent. *Knowl.-Based Syst.*, 13(5):285–296, 2000.

[43] Y. Li, C. Zhang, and S. Zhang. Cooperative strategy for web data mining and cleaning. *Applied Artificial Intelligence*, 17(5-6):443–460, 2003.

[44] 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.

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

[46] H. Lilian. An efficient sliding window algorithm for detection of sequential patterns, 2003.

[47] X. Ling, Q. Mei, C. Zhai, and B. Schatz. Mining multi-faceted overviews of arbitrary topics in a text collection. In *Knowledge discovery and data mining; KDD '08*, pages 497–505. ACM, 2008.

[48] B. Liu. *Web data mining : exploring hyperlinks, contents, and usage data*. Data-centric systems and applications. Springer, Berlin, 2007.

[49] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *Information and knowledge management; CIKM '09:*, pages 255–264. ACM, 2009.

[50] W. F. Madria S.K., Rhowmich S.S. Research issues in web data mining. In *In Proceedings of Data Warehousing and Knowledge Discovery*, pages 303–312, First International Conference, 1999.

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

[52] D. Metzler and W. B. Croft. Latent concept expansion using markov random fields. In *Research and development in information retrieval; SIGIR '07*, pages 311–318. ACM, 2007.

[53] morgan kaufmann. *Data Mining: Concepts and Techniques*. 2000.

[54] J. Mostafa, S. Mukhopadhyay, W. Lam, and M. J. Palakal. A multilevel approach to intelligent information filtering: Model, system, and evaluation. *ACM Trans. Inf. Syst.*, 15(4):368–399, 1997.

[55] N. Nanas, V. Uren, and A. de Roeck. A comparative evaluation of term weighting methods for information filtering. In *Proceedings on 15th International Workshop on Database and Expert Systems Applications.*, pages 13–17, 2004.

[56] N. Zhong, Y. Li, and S.-T. Wu. "Effective pattern discovery for text mining". *to appear in IEEE Transactions on Knowledge and Data Engineering (DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.211).*

[57] R. K. Pon, A. F. Cardenas, D. Buttler, and T. Critchlow. Tracking multiple topics for finding interesting articles. In *Knowledge discovery and data mining; KDD '07*, pages 560–569. ACM, 2007.

[58] J. M. Ponte. A language modeling approach to information retrieval. Master's thesis, 1998.

[59] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[60] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, and H. Li. Ranking with multiple hyperplanes. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 279–286. ACM, 2007.

[61] A. Rakesh and R. Srikant. Mining sequential patterns. In *In proceedings of the 11th International Conference on Data Engineering*, pages 3–14, 1995.

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

[63] S. Robertson and I. Soboroff. The trec 2002 filtering track report. *NIST Special Publication: SP 500-251 (TREC-2002)*, 2002.

[64] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 42–49. ACM, 2004.

[65] S. E. Robertson and I. Soboroff. The trec 2002 filtering track report. In *TREC*, 2002.

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

[67] S. E. Robertson, S. Walker, and M. Beaulieu. Experimentation as a way of life: Okapi at trec. *Inf. Process. Manage.*, 36(1):95–108, 2000.

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

[69] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System*, pages 313–323. Prentice Hall, 1971.

[70] T. Rose, M. Stevenson, and M. Whitehead. The reuters corpus volume 1 - from yesterdays news to tomorrows language resources. In *In Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31, 2002.

[71] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24:513–523, August 1988.

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

[73] R. E. Schapire, Y. Singer, and A. Singhal. Boosting and rocchio applied to text filtering. In *Research and development in information retrieval; SIGIR '98*, pages 215–223. ACM, 1998.

[74] S. Scott and S. Matwin. Feature engineering for text classification. In *The 16th International Conference on Machine Learning*, pages 379–388, 1999.

[75] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[76] M. Seno and G. Karypis. Slpminer: An algorithm for finding frequent sequential patterns using length-decreasing support constraint. *Data Mining, IEEE International Conference on*, 0:418, 2002.

[77] S. Shehata, F. Karray, and M. Kamel. A concept-based model for enhancing text categorization. In *Knowledge discovery and data mining; KDD '07*, pages 629–637. ACM, 2007.

[78] D. Shen, J.-T. Sun, Q. Yang, H. Zhao, and Z. Chen. Text classification improved through automatically extracted sequences. In *Data Engineering*, ICDE '06, pages 121–123. IEEE Computer Society, 2006.

[79] A. Smeaton and F. Kelledy. User-chosen phrases in interactive query formulation for information retrieval. In *iBCS Colloquium on Information Retrieval (IRSG98) (Springer-Verlag*, 1998.

[80] F. Song and W. B. Croft. A general language model for information retrieval. In *Information and knowledge management; CIKM '99:*, pages 316–321. ACM, 1999.

[81] E. R. Stephen and J. Karen Sparck. Relevance weighting of search terms. In *Document retrieval systems*, pages 143–160. Taylor Graham Publishing, 1988.

[82] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 675–684. ACM, 2004.

[83] X. Tao, Y. Li, and N. Zhong. A personalized ontology model for web information gathering. *Knowledge and Data Engineering, IEEE Transactions on*, 23:496 –511, april 2011.

[84] D. R. Tauritz, J. N. Kok, and I. G. Sprinkhuizen-Kuyper. Adaptive information filtering using evolutionary computation. *Inf. Sci.*, 122:121–140, 2000.

[85] J. Turmo, A. Ageno, and N. Català. Adaptive information extraction. *ACM Comput. Surv.*, 38:4, 2006.

[86] P. Tzvetkov, X. Yan, and J. Han. Tsp: Mining top-k closed sequential patterns. *Knowl. Inf. Syst.*, 7(4):438–457, 2005.

[87] D. W. Oard. The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, 7(3):141–178, 1997.

[88] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *Research and development in information retrieval; SIGIR '08*, pages 219–226. ACM, 2008.

[89] S.-T. Wu. *Knowledge discovery using pattern taxonomy model in text mining*. PhD thesis, Queensland University of Technology, 2007.

[90] S.-T. Wu, Y. Li, and Y. Xu. Deploying approaches for pattern refinement in text mining. In *Data Mining, 2006. ICDM '06. Sixth International Conference on*, pages 1157 –1161, 2006.

[91] S.-T. Wu, Y. Li, Y. Xu, B. Pham, and P. Chen. Automatic pattern-taxonomy extraction for web mining. *WI*, 00:242–248, 2004.

[92] J. Xu and W. B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, 18:79–112, 2000.

[93] Y. Xu and Y. Li. Generating concise association rules. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 781–790. ACM, 2007.

[94] Z. Xu and R. Akella. Active relevance feedback for difficult queries. In *Information and knowledge management; CIKM '08:*, pages 459–468. ACM, 2008.

[95] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Research and development in information retrieval, 2008*, SIGIR '08, pages 619–626. ACM, 2008.

[96] X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: a profile-based approach. In *Knowledge discovery in data mining; KDD '05*, pages 314–323. ACM, 2005.

[97] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large datasets. In *Data Mining (SDM03)*, pages 166–177, 2003.

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

[99] Y. Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1:69–90, 1999.

[100] Y. Yang and B. Kisiel. Margin-based local regression for adaptive filtering. In *Information and knowledge management; CIKM '03*, pages 191–198. ACM, 2003.

[101] Y. Yang, S. Yoo, J. Zhang, and B. Kisiel. 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*, pages 98–105. ACM, 2005.

[102] Y. Yang, S. Yoo, J. Zhang, and B. Kisiel. Robustness of adaptive filtering methods in a cross-benchmark evaluation. pages 98–105, 2005.

[103] M. J. Zaki. Spade: an efficient algorithm for mining frequent sequences. In *Machine Learning Journal, special issue on Unsupervised Learning*, pages 31–60, 2001.

[104] C. Zhai and J. Lafferty. Model-based feedback in the language modelling approach to information retrieval. In *Information and knowledge management; CIKM '01:*, pages 403–410. ACM, 2001.

[105] Y. Zhang. Using bayesian priors to combine classifiers for adaptive filtering. In *Research and development in information retrieval; SIGIR '04*, pages 345–352. ACM, 2004.

[106] Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *Research and development in information retrieval; SIGIR '01*, pages 294–302. ACM, 2001.

[107] Y. Zhang and J. Callan. Combining multiple forms of evidence while filtering. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 587–595. Association for Computational Linguistics, 2005.

[108] Q. Zhao and S. S. Bhowmick. Sequential pattern mining: A survey. Technical Report 2003118, CAIS, Nanyang Technological University, 2003.

[109] X. Zhou, Y. Li, P. Bruza, S.-T. Wu, Y. Xu, and R. Y. Lau. Using information filtering in web data mining process. In *Web Intelligence, IEEE/WIC/ACM International Conference on*, pages 163–169. IEEE Computer Society, 2007.

[110] X. Zhou, Y. Li, P. Bruza, and Y. Xu. "Integration of Information Filtering and Data Mining Process for Web Information Retrieval". 104-107, proceedings of the 12th Australian Document Computing Symposium (ADCS2007), 64-71, Melbourne, Australia, Dec. 2007.

[111] S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled classification using maximum entropy method. In *Research and development in information retrieval, 2005*, SIGIR '05, pages 274–281. ACM, 2005.