On-line Learning For Adaptive Text Filtering

YU Kwok Leung
Department of Systems Engineering & Engineering Management

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Systems Engineering and Engineering Management

June 1999

On-line Learning For Adaptive Text Filtering

Submitted by YU Kwok Leung

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in June 1999

Abstract

We investigate the unique requirements of the adaptive textual document filtering problem and propose a new on-line learning framework, known as the REPGER (RElevant feature Pool with Good training Example retrieval Rule) algorithm to tackle this problem. Our algorithm possesses three characteristics. First, it maintains a pool of selective features with potentially high predictive power to predict document relevance. Second, besides retrieving documents according to their predicted relevance, it also retrieves incoming documents which are considered as good training examples. Third, it can dynamically adjust the dissemination threshold throughout the filtering process so as to maintain a good filtering performance in a fully interactive environment. We have conducted experiments on three document corpora, namely, Associated Press, Foreign Broadcast Information Service and Wall Street Journal to compare the performance of our REPGER algorithm with two existing on-line learning algorithms. The results demonstrate that our REPGER algorithm gives better performance most of the time. Comparison with the TREC-7 adaptive text filtering track participants was also done. The result shows that our REPGER algorithm is comparable to them. Finally, we explore a technique for integrating a feature clustering method into our REPGER algorithm.

運用線上學習去處理自適應文件過濾問題

作者:庾國良

文摘

我們研究"自適應文件過濾"的獨特要求,從而提出一個新的線上學習框架名叫 REPGER 算法。我們的算法擁有三個特點。第一，它保持一個包含一些擁有高潛在預料能力的項的集，這個集是用來預料文件的關聯性。第二，除了根據文件的關聯性去過濾文件外，它亦會從大量輸入文件中自動收集一些有潛質的訓練例子。第三，它能動態地調整文件過濾的門限，使這算法能在互動環境下保持良好的性能。我們在三個文集上做了實驗，對我們的算法和另外兩個線上學習算法的性能作出比較。這三個文集分別是 Associated Press, Foreign Broadcast Information Service 和 Wall Street Journal。實驗結果顯示 REPGER 算法的性能在大部分的情況下都比較好。我們亦與 TREC-7 中自適應文件過濾組的參與者作出比較，結果顯示我們的算法與他們的不相伯仲。最後，我們探討一個技術去把項的聚集方法與 REPGER 算法結合。

Acknowledgments

I would like to take this opportunity to thank all those who have contributed to this thesis, directly or indirectly.

I would first like to express my sincere gratitude to my supervisor Prof. Wai Lam for his constructive criticisms, guidance and support and for sharing his immense wealth of knowledge, providing me the appropriate environment for my research, and being accessible at all times. I would also like to thank Prof. Kam-Fai Wong and Prof. Mei-Ling Meng for their valuable comments and constructive criticisms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Problem

The number of Internet users has been growing rapidly with the widespread use of personal computers and networks in recent years. This explosive growth has led to the growth in the amount of information resources available over the Internet. As more and more information becomes available electronically, it becomes increasingly difficult to search for information or to filter out non-interesting information from information streams for users. Therefore, it is critically important to develop effective filtering systems that help users decide which information is relevant to their preferences. We focus on textual information since texts are still the major form of information available.

An information filtering system is capable of automatically monitoring information sources to find documents for a particular information need. In practice, for each user, it starts with an initial filtering profile derived from the information need of the user. The system assists the user by filtering the information stream and delivering the relevant information to him/her. Users can optionally give feedback information to the information filtering system after reading the delivered documents. The feedback information is the evaluation given by the user on how relevant each document is for a specific information need. This feedback information is usually referred to as a *relevance judgment.* It can simply be a binary judgment indicating the relevance, i.e. being relevant or non-relevant, or a numerical score representing the likelihood of relevance. The system makes use of the relevance judgments to learn a more accurate filtering profile for each user. In summary, an information filtering system aims at learning a filtering model, which represents the information need given by the user, to filter information according to the filtering model.

1.2 Information Filtering

Information filtering deals with the delivery of information that is relevant to the user in a timely manner. Information Filtering (IF) and Information

Retrieval (IR) are two research fields addressing the problem of delivery of information to people who need it. At an abstract level, there is relatively little difference between IF and IR. Belkin and Croft [3] provided a detailed description of IF and identify the similarities and differences with IR. IF also shares some resemblance with text categorization [23]. Information Filtering can be classified into two settings, namely, batch filtering and adaptive filtering according to the filtering definitions in the Text REtrieval Conference (TREC) [1] [16, 17, 27]. Batch text filtering basically requires a training stage before starting the filtering process. In the training stage, the system takes as input a batch of training documents with user relevance judgments for a specific information need. The system is required to construct a filtering profile for the user's information need based on the set of training documents. This learning technique is called *off-line learning* which means that it can learn a filtering profile in a batch mode but not incrementally. After the filtering profile is learned, the system uses the profile to filter future documents for the user. The filtering profile usually remains unchanged during the filtering process. The batch filtering setting is summarized in Figure 1.1.

[1] The Text REtrieval Conference (TREC) was started in 1992 as part of the TIPSTER Text program. Its purpose is to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies.

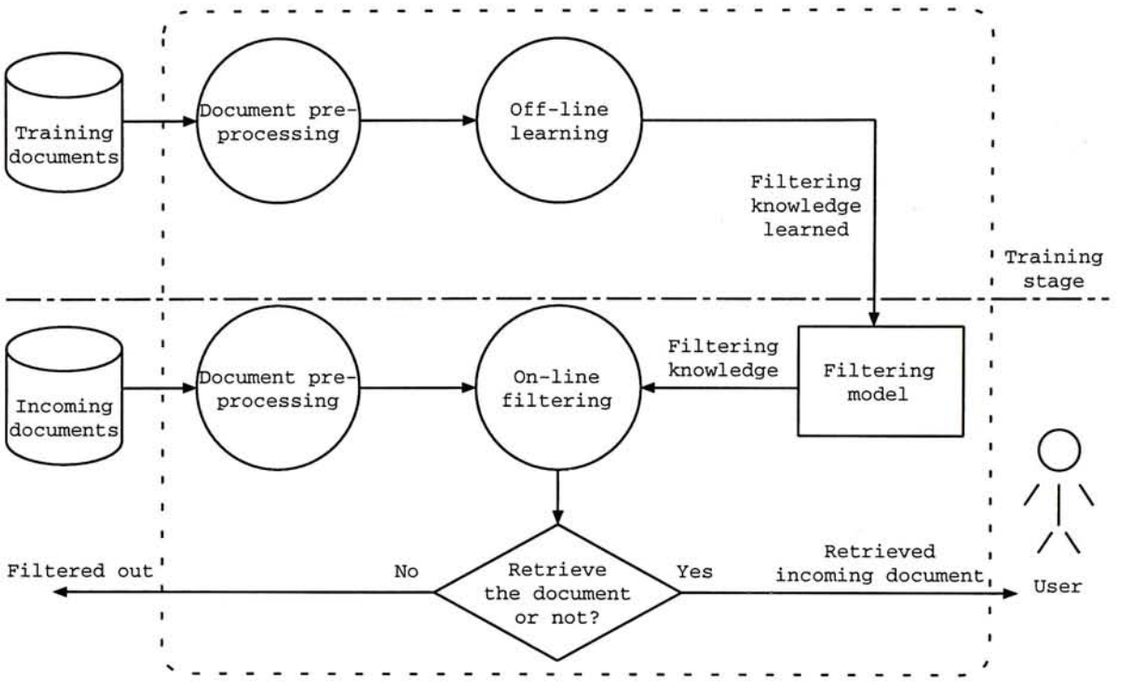

Figure 1.1: *Batch Filtering.*

As shown in Figure 1.2, the adaptive text filtering problem is designed to model a more realistic situation that a filtering system can only expect relevance judgments for documents which have been decided by the system for retrieving. The system starts only with the user information need, which is usually referred to as *topic description*, and no evaluated documents. It creates an initial filtering profile by using the description of the topic. Documents arrive sequentially in chronological order. The system analyzes the contents of the incoming documents. It makes use of the filtering profile to decide whether or not to retrieve each document based on the profile-document similarity. If a document is retrieved, the user can optionally provide a rel-

evance judgment of that document for the topic. The system can use the relevance information to update the filtering profile. This learning technique is known as *on-line learning* which means that it can learn a filtering profile incrementally in an interactive environment. This setting implies that relevance judgments from unretrieved documents are never revealed to the system. It does not have an explicit training stage as in batch filtering.

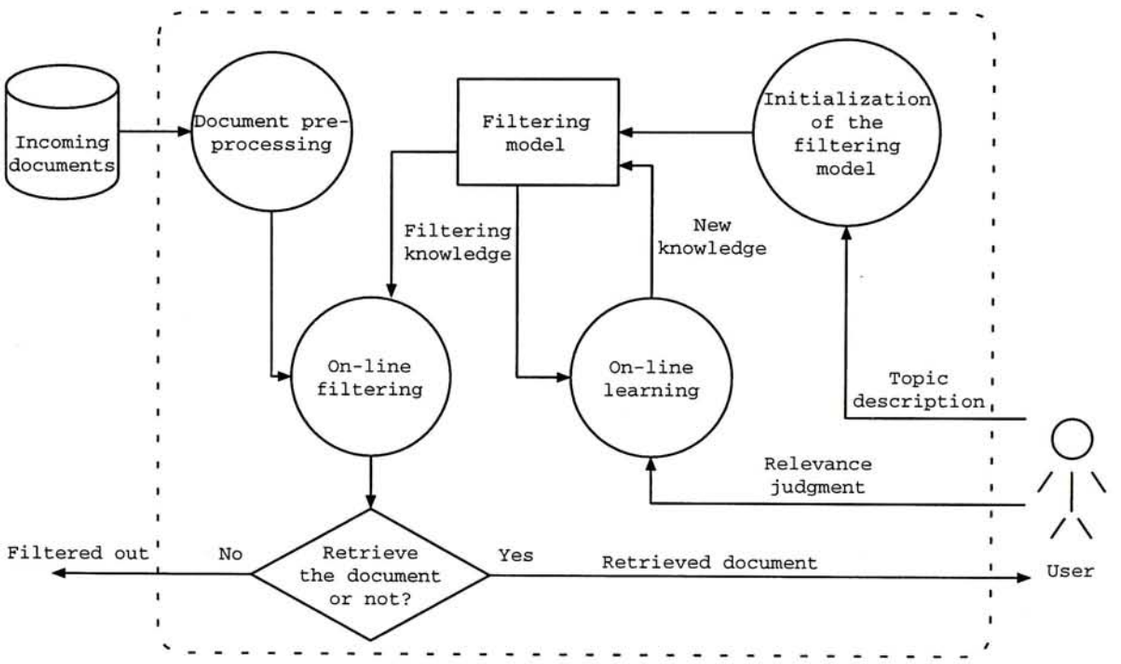

Figure 1.2: *Adaptive Filtering.*

In general, an adaptive information filtering system basically comprises four major components:

- **Filtering model:** It has a filtering model which stores the filtering knowledge of the information domain. The filtering knowledge actu-

ally represents how a relevant document should be. In other words, it represents the user's information need. The filtering model is usually referred to as a *filtering profile.*

- **Filtering profile initialization:** There is a filtering profile initialization mechanism. Before starting the filtering process, the user can provide his/her information need to the system so that the system can initialize its filtering profile according to the preference of the user. The system treats each user's information need as a topic. Therefore, this information need is usually referred to as a *topic description.*

- **Information filtering:** It has an information filtering mechanism to decide which documents should be treated as relevant and presented to the user. The criterion in predicting the relevance of the documents is based on the profile-document similarity.

- **On-line learning:** It has an on-line learning mechanism to update the filtering profile in order to improve future filtering performance. After reading each retrieved document, the user can provide a relevance judgment of the document to the system. The system can then learn a more accurate filtering profile by using the relevance judgment.

There are some advantages in adopting on-line learning technique of adaptive filtering over the off-line learning technique of batch filtering. First,

on-line learning does not need a large set of documents with relevance judgments to train the filtering profile. In fact, collecting a large set of documents with relevance judgments for each user's information need is practically impossible. Second, on-line learning is adaptable in an interactive environment. Since users' information needs may change or shift over time, on-line learning is capable of capturing the shift of users' preferences by learning from examples. This is achieved by learning incrementally from retrieved documents with relevance judgments.

1.3 Contributions

We have studied a number of existing information filtering systems. We discover that only some of them comprise all the four major components [2] that an adaptive filtering system should have. Since adaptive filtering is a new setting in information filtering introduced by TREC [16, 17, 27] in 1998, many filtering frameworks of the existing filtering systems were not designed to tackle the adaptive filtering problem. After investigating the existing filtering systems and some existing on-line learning algorithms, we find out three main tasks that an adaptive filtering system should be able to perform well in order to achieve good filtering performance. The three tasks

[2] Please refer to Section 1.2 for the details of the four components.

are: (1) Differentiating the features with potentially high predictive power from the other features so that the system can consider a set of informative features in predicting document relevance; (2) Solving the problem of lack of training examples (judged documents) in adaptive filtering; and (3) Varying the parameters used by an algorithm so as to maintain a good performance throughout the filtering process.

We propose an adaptive filtering framework based on on-line machine learning and content-based learning to tackle the adaptive text filtering problem [24, 44]. Our algorithm, known as the **REPGER (RElevant feature Pool with Good training Example retrieval Rule)** algorithm, possesses three characteristics. First, it maintains a pool of selective features with potentially high predictive power to predict document relevance. Second, it incorporates a novel mechanism for retrieving good training examples; this overcomes the problem of lack of training examples in an adaptive filtering environment. Third, it can dynamically learn the dissemination threshold, which is used to decide whether or not to retrieve an incoming document, so as to maintain a good filtering performance in an interactive environment. Specifically, the contributions made by this thesis are as follows:

- This thesis proposes a new on-line machine learning framework, known as the **REPGER** algorithm, to deal with adaptive filtering. New tech-

niques are developed for making the technical procedures of predicting document relevance and setting dissemination threshold more adaptable in an interactive environment of adaptive filtering. For predicting document relevance, a new concept, known as the **RElevant feature Pool (REP)**, is proposed to maintain a set of features with potentially high predictive power. For dissemination threshold setting, we propose a technique to adjust the threshold dynamically so as to maintain a good filtering performance throughout the filtering process.

- One unique characteristic of adaptive filtering is that there is no evaluated documents, i.e. documents with relevance judgment, for the system to learn the filtering profile before the filtering process. In the light of this characteristic, a new technique is proposed to tackle the problem of lack of training examples in the adaptive filtering setting which is known as the **Good training Example retrieval Rule (GER)**. The GER retrieves good training examples from the stream of incoming documents to help the system learn a more accurate filtering profile by using the relevance judgments of the retrieved documents.

- This thesis validates the use of the GER for tackling the problem of lack of training examples for the information filtering system in adaptive filtering.

- A new performance comparison method for information filtering systems is proposed for adaptive filtering. It reflects how well an information filtering system performs throughout the filtering process.

- Experimental results on three document corpora show that our REPGER algorithm is more effective than two existing on-line learning algorithms [7, 18] and the performance of our algorithm is comparable to the TREC-7 [17] adaptive filtering track participants.

- The feasibility of applying a feature clustering technique, known as the Distributional Clustering, in adaptive filtering is investigated. The experimental result shows that the effect of the clustering technique is quite promising.

1.4 Organization Of The Thesis

The rest of this thesis is organized as follows. Chapter 2 reviews some related work in information filtering. Chapter 3 presents the commonly used textual document representation techniques and analyzes two existing on-line learning algorithms and discusses their limitations. Some ideas of our new approach have been drawn from the analysis. Chapter 4 presents our REPGER algorithm which is a new on-line learning framework for adap-

tive text filtering. Chapter 5 describes a new threshold learning algorithm which is used in our REPGER algorithm for higher adaptability in filtering. Evaluation experiments and results are presented in Chapter 6. Chapter 7 describes the investigation of integrating REPGER with a feature clustering technique. Chapter 8 gives the concluding remarks and presents some directions for future work.

Chapter 2

Related Work

The work in this thesis is on the field of adaptive filtering which belongs to the research area of Information Filtering (IF). There are some well-developed information filtering systems employing different approaches to deal with IF.

The Stanford Information Filtering Tool (SIFT) was developed by Yan and Garcia-Molina [43]. It is basically an information retrieval system which does not support user profile learning. Users need to submit their profiles through a World Wide Web (WWW) browser, and then SIFT matches the users' profiles against the news articles. There are two alternative matching algorithms in SIFT, namely, the Boolean model and the vector space model [38]. A profile can be a Boolean conjunction of words for the Boolean model or a set of words and a relevance threshold for the vector space model. Matching articles are sent to the user by e-mail. The SIFT filtering engine

implements novel indexing techniques which are capable of scaling to large number of documents and profiles. The main disadvantage of SIFT is that it does not have any learning capability. If users are not satisfied with the retrieved articles or have different information needs, they need to manually re-construct their profiles until the retrieved articles satisfy them.

NewT is an information filtering system developed by Sheth and Maes [39, 40] and it is used for filtering Usenet news. It applies the vector space model in profile and document representations. It maintains a set of profiles for each user's information need. Documents are retrieved based on a combination of the predictions of the profiles in some manner. Existing profiles are learned from relevance feedback provided by the users. The system makes use of genetic algorithm (GA) to discover new profiles from existing ones and to flush out the unfit ones. One limitation of genetic algorithm is that it is unsuitable for adaptive filtering settings. Moreover, there is no mechanism to maintain the functional diversity of the profiles in the population.

InRoute is a document filtering system developed by Callan [6]. It makes use of the Bayesian inference network model [41] to process a query and a document. The major tasks performed by InRoute are creation of a query network (representing a user profile), creation of a document network (representing a document) and performing inference over the networks for filtering purpose. Users specify their information needs by using a query language or

in natural language and then InRoute transforms the description to a query network. The query language, which is also used in INQUERY [8], defines the syntax for users to specify their information needs. In order to save processing time, InRoute uses the "lightweight indexing" philosophy. Indexing speed is maximized by indexing terms that actually appear in one or more queries. A new inverted document frequency (IDF) estimation technique was also proposed. However, like SIFT, there is no mechanism for the system to update the query network by using feedback from the users.

Mostafa et al. [30] presented a general model of information filtering. As a way to reduce complexity, the architecture of the model incorporates multilevel functional decomposition and supports generality through modularity. A filtering system, namely SIFTER (Smart Information Filtering Technology for Electronic Resources), has been implemented based on the model. SIFTER employs established information retrieval and artificial intelligence techniques. They proposed to decompose the problem of learning a filtering profile into two levels. The top level represents a classification mapping from the document space to a finite number of classes. This mapping is learned in an off-line setting. The bottom level subsequently estimates the mapping describing user relevance for the different classes which can be done by on-line learning. The decomposition reduces the learning complexity but it limits the maximum achievable filtering accuracy for a class may not correspond

well to a user interest. Unlike many filtering systems, SIFTER does not make binary filtering decisions on incoming documents. It presents a number of ranked documents to users at a time.

Pazzani and Billsus proposed an extension of the World Wide Web (WWW) sites retrieval system, known as Syskill & Webert [32], to support revising of user provided profiles in [31]. Web pages are represented as a Boolean feature vector. Not all words that appear in an HTML document are used as features. The system uses an information-theoretic approach to determine which words to use as features. It employs a naive Bayesian classifier to revise profiles provided by users and to learn the profiles by using relevance feedback from users.

The systems described above apply a *content-based* approach to filter information. Information is filtered based on a comparison between its content and a user profile. Besides using the content-based approach, other systems apply a *collaborative* approach or a combination of them. In the collaborative approach, we filter information for a user by considering the filtering results of other similar users. Rather than computing the similarity of the information and the user profile in the content-based approach, we compute the similarity of the users.

Konstan et al. [19] discussed the challenges involved in creating a collaborative filtering system, named GroupLens, for Usenet news. The GroupLens

project was started in 1992 and completed a pilot study at two sites to establish the feasibility of using collaborative filtering for Usenet news [35]. The GroupLens server was a two-part database. The ratings database stores all ratings that users have given to articles. The correlations database stores information about the historical agreement of pairs of users. The prediction process reads both correlations and ratings and generates prediction. The problem of ratings sparsity in collaborative filtering is tackled by partitioning the set of Usenet news articles into clusters that are commonly read together.

Fab [2] is a distributed implementation of a hybrid content-based, collaborative Web page filtering system. It addresses the problem of how to combine both content-based and collaborative approaches in filtering Web pages. They maintain user profiles based on content analysis (content-based learning) and directly compare these profiles to determine similar users for collaborative filtering. Users receive items both when they score highly against their own profile, and when they are rated highly by a user with a similar profile. Users are required to assign appropriate ratings from a 7-point scale after reading the filtered Web pages. The users' ratings are used to update their profiles.

NewsWeeder is a netnews-filtering system proposed by Lang [25]. It addresses the problem of the reliance on the user for creating a user profile. The user can rate his or her interest level for each article being read, and

then NewsWeeder can learn a user profile based on these ratings. Currently, there are six interest levels: essential, interesting, borderline, boring, gong and skip. News articles and the user profile are represented by vectors composed of tokens as their elements. Besides words and a combination of words, tokens include punctuation and other specialized symbols also. The learning algorithm is based on the Minimum Description Length (MDL) principle. The MDL measure is used to find the best distribution of the tokens for each interest level. It provides an information-theoretic framework for balancing the tradeoff between model complexity and training error. NewsWeeder uses both content-based and collaborative filtering. The system uses the collected rating information to learn a new user profile each night. One deficiency of NewsWeeder is that it can only perform off-line learning.

After reviewing the above existing information filtering systems, we find that many of the above approaches are not effective for adaptive filtering. Several methods have been proposed to deal with adaptive filtering in the recent Seventh Text REtrieval Conference (TREC-7) [17] in 1998 which is the first year for TREC to organize the adaptive text filtering track. For instance, Eichmann et al. [12] developed a clustering method based on the standard cosine-similarity measure. There are two levels of clusters containing words derived from the topic description and the incoming documents. There are some thresholds used by the system for its dynamic clustering technique.

The primary cluster level corresponds to the internal representation of a topic description. Each primary cluster has a set of zero or more secondary clusters. When a document clears the threshold for a primary cluster, it either joins an existing secondary cluster or forms a new one, based upon a membership threshold. When a secondary cluster's similarity to a primary cluster exceeds a visibility threshold, its member documents are declared as relevant to the primary cluster. However, the binary decision to accept or reject a document is not made immediately until the similarity of the secondary cluster to the primary cluster exceeds the visibility threshold. This is an additional constraint imposed on the adaptive filtering setting.

Kwok et al. [22] conducted some experiments on applying the PIRCS system to perform adaptive filtering. The PIRCS system is based on the probabilistic indexing and retrieval models of [28, 36] but extended with the concept of document components [20, 21]. It is an information retrieval system designed to deal with information retrieval and batch filtering. They emphasize on dynamically setting a retrieval status value (RSV) threshold to select or not select a document for examination. A technique for adjusting the threshold was proposed. They implemented query weight adjustment only, but not query expansion. This helps the RSV's of documents remain in a stable range which would favor the performance of the threshold adjusting technique.

Zhai et al. [45] made use of the CLARIT system based on the simple Rocchio relevance feedback [37] to handle adaptive filtering. Each document is indexed on noun phrases and individual words using the standard CLARIT phrase indexing technique [13, 14, 29]. The filtering profiles and the documents are both represented as term vectors. Rocchio feedback, on relevant documents only, is used to expand the term vector. A method, known as the "delivery ratio" method, is used to estimate an initial profile threshold. The delivery ratio equals to the desirable number of documents to deliver over the total number of documents. A small reference corpus can be used to estimate an approximate threshold score at which the desirable ratio would be achieved. However, the accuracy of the initial threshold estimated on the reference corpus is not guaranteed. This is because different topics and corpora may have different characteristics and score ranges. Besides estimating the initial threshold, they also proposed a technique to adjust the threshold dynamically.

We have studied a number of existing information filtering systems above. Since adaptive filtering is a new setting in information filtering introduced by TREC [16, 17, 27] in 1998, many filtering frameworks of the existing filtering systems were not designed to tackle the adaptive filtering problem. We discover that only some of them comprise all the four major components [1]

[1] Please refer to Section 1.2 for the details of the four components.

that an adaptive filtering system should have. SIFT [43] and InRoute [6] do not support user profile learning. NewsWeeder [25] can only performs off-line learning. Even some of them can be potentially applied in adaptive filtering, they still have deficiencies. NewT [39, 40] applies GA to generate new user profiles but there is no mechanism to maintain the functional diversity of the profiles. SIFTER [30] decomposes the procedure of profile learning into two levels in order to reduce learning complexity, however, it limits the maximum achievable filtering accuracy. GroupLens [19], a collaborative filtering system, is not effective for adaptive filtering because the problem of lack of training examples in adaptive filtering makes the system hard to create filtering profiles for users especially when the user's preference is very different from the others. For the TREC-7 adaptive text filtering track participants' systems, most of them are not originally designed for adaptive filtering. For example, PIRCS [22] is designed to deal with information retrieval and batch filtering. They only adjust the retrieval status value (RSV) threshold in order to make the system suitable for adaptive filtering. The other systems also have deficiencies as we mentioned above.

In general, the suitability of a filtering system for adaptive filtering mainly depends on the learning algorithm used by the system. One of the four major components that an adaptive filtering system should have is *On-line Learning*. We believe that adaptive filtering can be effectively tackled by on-

line machine learning techniques. Therefore, we investigate two existing on-line learning algorithms. Recently Cohen and Singer developed the sleeping experts algorithm [10] for performing automatic text categorization. This on-line learning algorithm can potentially be applied to adaptive filtering. Callan [7] proposed an improved Exponentiated-Gradient (EG) algorithm, which is also an on-line learning algorithm, to solve the adaptive filtering problem. We present a more detailed analysis of these two algorithms and their shortcomings in Section 3.2.

Chapter 3

Adaptive Text Filtering

3.1 Representation

Unstructured textual documents and topic descriptions must be represented in a way which can be processed by information filtering systems. We describe a representation used in many filtering systems including our approach. The representation is based on the vector space model [38]. The vector space model assumes that a feature set is available to identify both documents and topic descriptions. Both filtering profiles and documents can then be represented as *feature vectors* in some hyper-space. A distance metric which measures the proximity of vectors to each other is defined over the space. Hence, the profile-document similarity can be computed by using the distance metric. The advantage of using the vector space model is its flexibility - as

such a document can also be represented as a topic description. Users can provide samples of interesting documents as an alternative to constructing the filtering profile. The representations used for documents and filtering profiles are described below.

3.1.1 Textual Document

In textual document representation, appropriate words or phrases are extracted from a document to form a vector representing the document. A textual document needs to be preprocessed before the extracted words or phrases can be used in document representation. There are three basic steps in document preprocessing. The first step is to remove the punctuation marks and to change all alphabets to lower cases. The second step is to remove non-informative words or common function words (stop-words) such as "I", "an", "of" and "but". These words are eliminated for they are useless in content identification of the textual documents. This step is usually referred to as *stop-word removal.* The third step is to conduct *word stemming* which is a process to transform all words to their word-stems. The purpose of word stemming is to make sure that words which differ only in tenses or part of speech can be matched. For example, the words "looking" and "looks" would be transformed to "look". After punctuation mark removal, lower case

transformation, stop-word removal and word stemming, the selective words or phrases generated from the remaining words are called *features* or *terms*, which are used to characterize the document.

Here is an example which shows the procedures of document preprocessing of an original document. The raw text is quoted from an article of the Associated Press document corpus.

- *Raw text:*

  Brezhnev ruled for 18 years before he died in 1982. After his death a city, streets, city squares and state-run enterprises were named after him. Soviet officials started removing those names in January.

- *Removing punctuation marks and changing all alphabets to lower case:*

  brezhnev ruled for 18 years before he died in 1982 after his death a city streets city squares and state-run enterprises were named after him soviet officials started removing those names in january

- *Stop-word removal:*

  brezhnev ruled years died 1982 death city streets city squares state-run enterprises named soviet officials started removing names january

- *Word stemming:*

brezhnev rule year di 1982 death citi street citi squar state-run enterpris name soviet offici start remov name januari

The word stemming algorithm used here was proposed by Porter in [34].

Since the features are not equally important for content representation, *document weights* are assigned to the features according to their relative importance. TF-IDF is one of the commonly used document weights of a feature. It is the product of the term frequency (TF) in a document and the inverted document frequency (IDF) in a document collection. The term frequency of a feature is the occurrence frequency of the feature in a particular document and reflects the importance of the feature in that document. The inverted document frequency of a feature is a factor reflecting whether the feature is document-specific or not. It attains a high value if the feature appears in fewer documents. A commonly used measure for the inverted document frequency of a feature f_k is

$$\text{IDF}_k = \log(\frac{N}{n_k})$$

where N is the total number of documents that the system has encountered, and n_k is the number of documents that contain the feature f_k. Note that the inverted document frequency must be calculated incrementally in the adaptive filtering environment. Consider a document D. The document weight of the feature f_k appearing in the document D is given as

$$x_k = \mathrm{TF}_k \times \mathrm{IDF}_k$$

where TF_k is the term frequency of the feature f_k in the document D, and IDF_k is the inverted document frequency of the feature f_k in the document collection. Consequently, the document D can be represented as a vector of features

$$\vec{D} = < x_1, \ldots, x_j, \ldots, x_n > \tag{3.1}$$

where n is the total number of distinct features in the document collection, and x_j is the document weight of the feature f_j in document D. Sometimes, we normalize this vector to become a unit vector before it is used for subsequent processing.

The number n is not known in adaptive filtering. To handle this situation, a document could be represented as a *set* of features

$$\mathcal{D} = \{f \mid f = f_i, i = 1, 2, \ldots, l\} \tag{3.2}$$

where l is the number of distinct features in the document D. Each feature is associated with its document weight. When we process a document, we usually only need the document weight of a feature in the document. Therefore, this set-based representation essentially embodies the same information

needed for processing a document as in the vector-based representation.

Note that the topic description provided by the user for the filtering system to initialize the filtering profile is also a textual document. We can apply the technique mentioned above to represent the topic description. However, there is no statistics to calculate the IDF of the features extracted from the topic description in the adaptive filtering setting. We can solve this problem by using two alternatives. One is to use the TF of the features as their document weights, i.e. $x_k = \mathrm{TF}_k$, or to use a binary feature representation, i.e. the document weight of a feature is 0/1 if the feature is absent/present in the document. The second alternative is to calculate the IDF of the features by using the statistics collected from another unrelated document corpus, known as the *reference corpus*. This technique can also be used in calculating the IDF of the features of the incoming documents. The IDF of a feature f_k calculated by using this technique is given as

$$\mathrm{IDF}_k^{ref} = \log(\frac{N^{ref} + N}{n_k^{ref} + n_k}) \tag{3.3}$$

where N^{ref} is the total number of documents in the reference corpus, and n_k^{ref} is the number of documents in the reference corpus that contain the feature f_k.

3.1.2 Filtering Profile

The representation of a filtering profile is similar to that of a document. However, there are some differences between filtering profile and document representations. First, a filtering profile stands for a user's information need. Each feature in the profile vector is associated with a non-negative *feature weight* indicating the relative importance of the feature for the user's information need. Consequently, the filtering profile can be represented as a vector of features

$$\vec{W} = < w_1, \ldots, w_j, \ldots, w_n > \tag{3.4}$$

where n is the total number of distinct features in the document collection, and w_j represents the feature weight of the feature f_j in the filtering profile. Like document representation, the total number of distinct features n is not known in the adaptive filtering setting. Similar to Equation 3.2, the filtering profile can also be represented as a set of features

$$\mathcal{W} = \{f \mid f = f_i, i = 1, 2, \ldots, p\} \tag{3.5}$$

where p is the number of features in the filtering profile. Each feature is associated with its feature weight.

3.2 On-line Learning Algorithms For Adaptive Text Filtering

After a document is converted into an internal representation, we can employ some existing on-line learning algorithms for conducting adaptive filtering. Two on-line learning algorithms, namely, the sleeping experts algorithm [10] and the Exponentiated-Gradient (EG) algorithm [18] are investigated. Both algorithms tackle filtering by using a common two-step procedure similar to the one used in many other information filtering systems. For each profile-document pair, a document relevance score is first calculated and a dissemination threshold is applied to make the binary decision to accept or reject the document. Both algorithms use multiplicative update techniques to update their filtering profiles. The analysis of these two algorithms are presented below.

3.2.1 The Sleeping Experts Algorithm

The sleeping experts algorithm has recently been applied to conduct automatic text categorization [10]. However, it can potentially be employed to solve adaptive filtering. It associates an "expert" with each distinct feature to predict the relevance of incoming documents. As shown in Figure 3.1, it has a master algorithm which updates the weight of each expert using multi-

plicative update and combines the predictions of the experts. Motivated by the Infinite Attribute Model [4], it does not need to know the vocabulary in advance. Each expert consists of two "mini-experts". The first mini-expert, $\tilde{f}_1$, consistently predicts that the document is relevant whenever the corresponding feature is present in the document. The second mini-expert, $\tilde{f}_0$, consistently predicts that the document is non-relevant whenever the corresponding feature is present in the document.

A good adaptive text filtering algorithm should give a good performance not only at the end of the filtering process, but also during the filtering task. However, the sleeping experts algorithm may give undesirable performance before it has processed enough documents to allow the weights of the mini-experts to converge. If an expert appears in some non-relevant documents, this expert will have a negative predictive power, i.e. the weight of $\tilde{f}_1$ is smaller than that of $\tilde{f}_0$. However, if this expert appears in the next incoming document and the document is relevant, it will reduce the degree of the predicted relevance for that document, i.e. reduces the score of the function in Step 4 in Figure 3.1. Experts having negative predictive power are likely to mis-predict the relevance of the documents. Therefore, the idea of using "mini-expert" may not be effective for adaptive filtering especially during the initial period. Another drawback is that the sleeping experts algorithm treats every new feature as an expert (see Step 2, Figure 3.1). It means

Parameters: $\alpha \in (0,1)$, $\theta \in (0,1)$.

Initialize: $Pool \leftarrow \emptyset$

Do while there is an incoming document

1. Receive a new document $\mathcal{D} = \{f \mid f = f_i, i = 1, 2, \ldots, l\}$ and its classification $y_D \in \{0,1\}$.

2. Define the set of active mini-experts:

$$\mathcal{E} = \{\tilde{f}_m \mid f \in \mathcal{D}, m \in \{0,1\}\}$$

3. Initialize the weights of new mini-experts:

$$\forall \tilde{f}_m \in \mathcal{E} \;\; s.t. \;\; \tilde{f}_m \notin Pool : p_{\tilde{f}_m} = 1$$

4. Classify the document as positive if

$$\frac{\sum_{f \in \mathcal{D}} p_{\tilde{f}_1}}{\sum_{f \in \mathcal{D}} \sum_{m=0,1} p_{\tilde{f}_m}} > \theta$$

5. Update weights:

$$p_{\tilde{f}_m} = \begin{cases} p_{\tilde{f}_m} & y_D = m \\ \alpha p_{\tilde{f}_m} & y_D \neq m \end{cases}$$

6. Re-normalize the weights of the mini-experts.

7. Update: $Pool \leftarrow Pool \cup \mathcal{E}$.

Figure 3.1: *The Sleeping Experts Algorithm.*

that even the features appearing only in non-relevant documents have the same degree of contribution as relevant features in predicting the document relevance. However, it is less likely to have regularities for the features which appear only in non-relevant documents in text filtering. We think it is more appropriate to have different treatments on these two kinds of features.

Besides having drawbacks in tackling adaptive filtering, another deficiency of the sleeping experts algorithm is that it has no filtering profile initialization mechanism to process topic descriptions provided by the users.

3.2.2 The EG-based Algorithms

The Exponentiated-Gradient (EG) algorithm [18] has been applied to tackle adaptive filtering in [7]. The EG algorithm is designed for on-line prediction problems. It maintains a linear weight vector $\vec{W}$ as described in Equation 3.4 with each component representing a non-negative feature weight of each distinct feature being considered [1]. The sum of all components of the vector should be one. A document D is represented by using the vector space model as described in Equation 3.1.

The EG algorithm updates its linear weight vector by using a weight update formula when an incoming document is retrieved, i.e. it is predicted

[1]The dimension of the weight vector can be dynamically adjusted by incorporating the Infinite Attribute Model [4] into the EG algorithm.

as relevant by the system. The weight update formula for the linear weight vector $\vec{W}$ is given as follows:

$$w_j = \frac{w_j \exp(-2\eta(\vec{W} \cdot \vec{D} - y_D)x_j)}{\Sigma_{j=1}^{n} w_j \exp(-2\eta(\vec{W} \cdot \vec{D} - y_D)x_j)} \tag{3.6}$$

where $\vec{D}$ is a vector representing the incoming document D (usually a unit vector with elements represented as TF-IDF); y_D represents the target value of the classifier for the incoming document D; and η is the learning rate used to determine how rapidly the classifier learns from each retrieved document. Kivinen and Warmuth [18] suggested that the learning rate should be determined by the formula $\eta = 2/(3R^2)$ where R is a value that satisfies the constraint $\max(\max_j x_j - \min_j x_j) \leq R$ [2]. Initial weights of all features are usually set to $1/n$. At any time instance, the classifier predicts an incoming document as relevant when the inner product $\vec{W} \cdot \vec{D}$ exceeds a dissemination threshold. The document relevance prediction rule of the EG algorithm is shown below:

$$\text{If } \vec{W} \cdot \vec{D} \geq \phi, \text{ then retrieve that document}$$

where ϕ is the dissemination threshold.

Although the EG algorithm does not have features with negative predictive power, it may also give undesirable performance before it processes

[2] Refer to as KW in this thesis.

enough documents to allow the weights of the linear weight vector to converge. To illustrate this point, let us consider the following cases. Assume we begin with a uniform initial weight for each feature in the weight vector. Suppose a small learning rate η is used. If some features appear only in the non-relevant documents in the document corpus, the feature weights of these features in the weight vector will still be quite high after the algorithm learns from some documents. It is due to a low learning speed when a small learning rate is used. At this time instance, the algorithm maintains relatively high predictive power for these features. However, in fact, features which appear only in the non-relevant documents are less likely to have high predictive power. These unnecessarily high weights will lead to a poor filtering performance. Conversely, suppose a high learning rate η is used. If some features appear in both the relevant and the non-relevant documents and these documents arrive in a random order. The weights of these features will likely increase and decrease alternatively during the filtering process. Essentially, it leads to weight fluctuation which in turn will affect effectiveness throughout the filtering process. In conclusion, the above drawbacks are caused by a similar problem faced by the sleeping experts algorithm [10]. This is the problem of not differentiating features appearing only in non-relevant documents. Moreover, there is also no filtering profile initialization mechanism in the EG algorithm.

In [7], the EG algorithm was improved by addressing the problems of adjusting the target values and the dissemination threshold. It was proposed that the target values should be adjusted to the minimum and maximum document relevance scores that the current filtering profile can give. The minimum and maximum document relevance scores given by the filtering profile are proved to be the minimum and maximum document weights of the features in the current incoming document (called MinMax in this thesis). Therefore, the target values at a time instance should be the minimum and maximum document weights of the features in the current incoming document instead of using fixed target values throughout the whole filtering process. However, according to the on-line learning framework of the EG algorithm, the target values should remain constant since the target values represent the target levels of the inner product $\vec{W} \cdot \vec{D}$ for relevant and non-relevant documents. The goal of the on-line learning algorithm is to adjust the weights in the weight vector so that the error between the target levels and the inner product is minimized. If we apply the proposed method in [7], the target values will be dependent on the current incoming document. In other words, the target levels of the filtering profile are restricted by different incoming documents at different time instances. The scores given by the filtering profile for relevant and non-relevant documents will hardly reach the real target levels. Hence, the target levels should not be varied for different

incoming documents.

Another improvement suggested in [7] is to adjust the threshold of the classifier during the filtering process so that the filtering system can find a good threshold dynamically. The threshold is set somewhere between the average score of the retrieved relevant documents and the average score of the retrieved non-relevant documents. In order to reduce the risk of setting a poor threshold, the system can start to adjust the threshold after retrieving some relevant and non-relevant documents.

Chapter 4

The REPGER Algorithm

4.1 A New Approach

After investigating the existing filtering systems and some existing on-line learning algorithms, we find out three main tasks that an adaptive filtering system should be able to perform well in order to achieve good filtering performance. The three tasks are: (1) Differentiating the features with potentially high predictive power from the other features so that the system can consider a set of informative features in predicting document relevance; (2) Solving the problem of lack of training examples (judged documents) in adaptive filtering; and (3) Varying the parameters used by an algorithm so as to maintain a good performance throughout the filtering process.

We propose a new content-based on-line learning algorithm, known as

the **REPGER (RElevant feature Pool with Good training Example retrieval Rule)** algorithm, for the adaptive text filtering problem. The filtering process typically starts with an optional description of the topic or information need. When there is an incoming document, the system computes a document relevance score and then apply a dissemination threshold to make the binary decision to accept or reject the document. Our learning algorithm maintains a pool of selective features with potentially high predictive power to predict document relevance and adjusts the dissemination threshold dynamically so that the filtering system is more adaptable in an interactive environment. In adaptive filtering, one of the most challenging problems is the lack of training examples for the system to learn an accurate filtering profile. In the light of this problem, we propose a novel mechanism to retrieve good training examples at the beginning stage of the filtering process. In summary, there are three characteristics in our REPGER algorithm shown as follows:

- Predicting the likelihood of relevance of incoming documents by maintaining a pool of selective features with potentially high predictive power.

- Retrieving incoming documents based on the merit as good training examples to tackle the problem of lack of training examples in adaptive

filtering.

- Adjusting the dissemination threshold dynamically so as to maintain a good performance of the classifier during the filtering process.

We present the whole REPGER algorithm in Figure 4.1. Each characteristic of the algorithm is explained in detail in the following subsections. Appropriate features are automatically extracted from the documents in the preprocessing stage. A document is represented by a set of features extracted from the document and each feature is associated with a *document weight* as described in Equation 3.2. Two sets of features, $Pool_R$ and $Pool_N$, are maintained throughout the filtering process. $Pool_R$ contains features from the topic description and the retrieved relevant documents while $Pool_N$ contains features appearing only in retrieved non-relevant documents. Therefore, the two sets are complement and contain the features of all retrieved documents and the topic description. A *feature weight* is associated with each feature in $Pool_R$ and $Pool_N$. The feature weights of the features in $Pool_R$ are used in predicting document relevance. $Pool_R$ stands for the filtering profile and the representations of $Pool_R$ and $Pool_N$ are equivalent to the one described in Equation 3.5. Once an incoming document in retrieved, either or both of the two sets will be updated depending on the relevance judgment of the retrieved document. When the retrieved document is relevant, all the

features of the document which are not contained in $Pool_R$ are moved into $Pool_R$. If the features are new features, i.e. the features which never appear in retrieved documents, they associated with their initialized feature weights will be moved into $Pool_R$ directly. If the features are in $Pool_N$, they associated with their feature weights in $Pool_N$ will be moved into $Pool_R$. When the retrieved document is non-relevant, the new features are moved into $Pool_N$ with their initialized feature weights. After an incoming document is retrieved, the feature weights of the features in $Pool_R$ and $Pool_N$ are normalized separately so that they are summed to one in each set. The normalized weight is the unnormalized weight over the sum of the unnormalized weights. There is a feature weight allocation component in Step 4(a) in Figure 4.1 of our REPGER algorithm. The weight allocation component can be implemented by a variety of weight updating techniques. We use the weight update formula of the EG-based algorithm described in Equation 3.6 in our current implementation so that we have a fair performance comparison with the EG-based algorithms in the experiments.

Definition:
$Pool_R$ - a set of features appearing in retrieved relevant documents and topic description,
$Pool_N$ - a set of features appearing only in retrieved non-relevant documents,
f_k - the k-th feature that the system has encountered,
x_k - the normalized document weight of f_k that appears in the incoming document,
w_k - the normalized feature weight of f_k that appears in the incoming document.

Initialization: extract features $\mathcal{Q} = \{f_1, f_2, ..., f_q\}$ from the topic and initialize their feature weights. $Pool_R \leftarrow \mathcal{Q}$. $Pool_N \leftarrow \emptyset$

Do while there is an incoming document

1. Let $\mathcal{D}$ be the set of features extracted from the document. Calculate the normalized document weight for each feature in $\mathcal{D}$. Initialize the feature weights of the new features, i.e. $f_k \notin (Pool_R \cup Pool_N)$, to be $1/|Pool_R|$.

2. If $\sum_{f_k \in (\mathcal{D} \cap Pool_R)}(x_k w_k) = score_D \geq \phi$, then retrieve the document

3. If $\frac{|\mathcal{D} \cap Pool_R|}{|Pool_R|} + \frac{|\mathcal{D} - (Pool_R \cup Pool_N)|}{|\mathcal{D}|} \geq \beta$, then retrieve the document

4. If the document is retrieved in Step 2 or 3,

   (a) Get the classification (target value) y_D of the document and update the feature weights of the features in $\mathcal{D}$ by:

   $$w_k = w_k \exp(-2\eta(score_D - y_D)x_k)$$

   where η is the learning rate

   (b) Update $Pool_R$ and $Pool_N$ and re-normalize the feature weights of the features in them separately so that they are summed to 1.

   (c) If the number of retrieved documents is greater than a predefined number, then invoke the threshold learning algorithm.

Figure 4.1: *The REPGER Algorithm.*

4.2 Relevance Prediction By RElevant feature Pool

As discussed in Section 3.2, not all the features that appear in incoming documents should have the same degree of contribution in predicting document relevance. In other words, some features may not be essential for the system to consider in predicting document relevance. In batch filtering, a filtering profile usually considers all distinct features in the document collection. Deciding which feature to be considered by the filtering profile is not a main concern. It is because there is a large set of training documents for the system to train an accurate filtering profile with appropriate feature weight for each feature in the profile. However, deciding which feature should be considered by a filtering profile is a critical problem in adaptive filtering. The filtering profile is learned incrementally while it is being used to filter incoming documents. If the filtering profile considers too many features which are actually without significant predictive power, the performance of the system will be affected badly until the feature weights of these features converge to a negligible level.

Our REPGER algorithm carefully chooses the appropriate features being considered in predicting the likelihood of relevance of incoming documents. Inspired by the idea of Infinite Attribute Model [4], our algorithm does not

need to specify the full feature set in advance [1]. The Infinite Attribute Model defines a model of document and filtering profile representations which is different from the vector space model. For the vector space model, the dimension, which is the total number of distinct features in the document corpus, of the feature vector used for representation must be known in advance. In contrast, the Infinite Attribute Model represents each document and filtering profile by using a set of features. The size of the set depends on the number of features in the documents or the filtering profile. Hence, documents and filtering profile can be represented by using the Infinite Attribute Model without the requirement of knowing the total number of distinct features in the document corpus in advance. Therefore, this model is suitable for on-line learning algorithm in adaptive filtering. We observe that the features which appear only in non-relevant documents are less likely to have high predictive power because it is less likely to have regularities for these features. In contrast, the features that appear in the topic description or in relevant documents have potentially high predictive power because they are potentially related to the topic. In view of this observation, we make an assumption about the characteristic of the features. The assumption is that the features appearing in the topic description and the relevant documents are essential

[1] The REPGER algorithm uses Equations 3.2 and 3.5, which apply the Infinite Attribute Model, in document and profile representations respectively.

for the filtering system to consider in predicting document relevance. We are not going to claim that the features appearing only in non-relevant documents are not informative. We just think that they may not help much in document relevance prediction. Based on this assumption, we maintain a **RElevant feature Pool (REP)** containing features from the topic description and the retrieved relevant documents. This pool is used for predicting the likelihood of relevance of incoming documents. Suppose f_k denotes the k-th feature that the system has encountered. Let x_k and w_k be the document weight and the feature weight of f_k respectively. Let $\mathcal{D}$ be a set of features of the incoming document and $Pool_R$ be a set of features that appear in the topic description or in the previously retrieved relevant documents. The relevance prediction rule is:

$$\text{If} \sum_{f_k \in (\mathcal{D} \cap Pool_R)} (x_k w_k) \geq \phi, \text{ then retrieve that document}$$

where ϕ is the dissemination threshold.

The features in $Pool_R$ will be updated throughout the filtering process as shown in Step 4(b) in Figure 4.1. When the retrieved document is relevant, the features of the retrieved relevant document which are not in $Pool_R$ will be moved into $Pool_R$. This means that the size of $Pool_R$ is monotonically increasing. The features that appear only in non-relevant documents will not affect the predicted document relevance because all features in $Pool_R$

come from the topic description and the retrieved relevant documents. This design is more effective than the EG-based algorithms [7, 18] and the sleeping experts algorithm [10] for adaptive text filtering problems as shown in the experimental results (see Section 6.3). The REP design does not apply the concept of mini-expert. Therefore, it will not face the problem of negative predictive power of the sleeping experts algorithm [2]. Moreover, it will not face the problem of learning rate setting of the EG algorithm [3]. This is because the possibility of having the problem of low learning rate setting is much lower than the EG algorithm when we apply the REP design in predicting document relevance. Hence, we can set the learning rate at a value for a desirable learning speed without worrying the problems faced by the EG algorithm.

4.3 Retrieving Good Training Examples

The relevance prediction rule concentrates on retrieving incoming documents based on the likelihood of relevance. The documents predicted as relevant by the relevance prediction rule are presented to the user and then the relevance judgments from the user are used to update the filtering profile. Every retrieved document is actually a training example for the system to learn

[2]Please refer to Section 3.2.1.

[3]Please refer to Section 3.2.2.

a more accurate filtering profile. If no incoming document is predicted as relevant by the relevance prediction rule, there will be no training example (retrieved document) for the system to update the filtering profile. Therefore, in order to have enough training examples for the system to learn, we have to retrieve an incoming document not only according to its predicted degree of relevance, but also according to its value as a training example to improve the future filtering performance.

We may simply set a lower dissemination threshold at the beginning stage of the filtering process to let the system retrieve more documents for it to learn. However, this strategy only makes use of $Pool_R$, i.e. the filtering profile, to retrieve incoming documents. If the features in $Pool_R$ are not informative owing to a poor topic description, the documents retrieved by using this strategy will not help much in improving the future filtering performance. We observe that if we retrieve an incoming document which contains many new features, i.e. the features which never appear in retrieved documents, and many features in $Pool_R$, this document can help us learn more new features. It also allows the weight allocation component to update the weights of the features in $Pool_R$ by using the relevance judgment of the retrieved document to improve the future filtering performance. In view of this observation, we make an assumption about the characteristic of good training examples in adaptive filtering. The assumption is that an incoming

document is a good training example if it contains many new features and many features in $Pool_R$. Therefore, we introduce the **Good training Example retrieval Rule (GER)** to retrieve the incoming documents which have many new features and many features in $Pool_R$. Note that the GER are not designed to retrieve relevant documents. The task of retrieving relevant documents is done by the document relevance prediction rule presented in Section 4.2. The purpose of the GER is to retrieve documents which meet the criteria of being a good training example to improve the accuracy of the filtering profile by using the relevance judgments of the retrieved documents. In order to achieve this purpose, we need to maintain another set of features to keep track of the features of the previously retrieved documents. Let $Pool_N$ be a set of the features which appear only in previously retrieved non-relevant documents. The GER is:

$$\text{If } \frac{|\mathcal{D} \cap Pool_R|}{|Pool_R|} + \frac{|\mathcal{D} - (Pool_R \cup Pool_N)|}{|\mathcal{D}|} \geq \beta, \text{ then retrieve that document}$$

Similar to $Pool_R$, the features in $Pool_N$ will be updated throughout the filtering process.

Obviously, when considering the two fractions in the condition part of the GER independently, the maximum score given by each fraction is one. The overall score range given by the GER is [0,2). By setting the value of β higher than one, only the incoming documents satisfying our good training example

criteria, i.e. containing many new features and many features in $Pool_R$, will be retrieved be the GER. The higher is the value of β, the more strict the requirement of being a good training example is. The purpose of the GER is to retrieve a small number of good training examples for the system to learn at the beginning stage of the filtering process. Therefore, the parameter β should not be too small. A good strategy is to set β to a value slightly lower than the value at which the GER gives no effect. Moreover, the effect of the GER is continuously decreasing during the filtering process because the number of new features is decreasing and the number of features in $Pool_R$ is increasing as the system is retrieving more and more incoming documents. In other words, given a fixed value of β, the possibility of retrieving documents by the GER reduces during the filtering process. It means that the requirement of being a good training example set by the GER is adaptable in an interactive environment. This characteristic of the GER ensures that the GER will only retrieve a small number of good training examples.

4.4 Learning Dissemination Threshold Dynamically

In our REPGER algorithm, one characteristic is that the dissemination threshold can be adjusted dynamically throughout the filtering process. In an adaptive filtering environment, the filtering knowledge of the filtering model is changing all the time. Therefore, the dissemination threshold, which is used to decide whether or not to retrieve incoming documents, should be varied dynamically in order to make the filtering system more adaptive and effective. We propose to learn a local optimal dissemination threshold with respect to the performance measure dynamically during the filtering process. Our threshold learning algorithm is fully automatic which does not need any user intervention. Chapter 5 describes our threshold learning algorithm in detail.

Chapter 5

The Threshold Learning Algorithm

5.1 Learning Dissemination Threshold Dynamically

The performance of an information filtering system, which makes use of a dissemination threshold to decide whether or not to retrieve incoming documents, is heavily dependent on the setting of the dissemination threshold. In adaptive text filtering problems, the optimal value of the dissemination threshold for the classifier is unknown before the filtering process. Even if we can estimate the optimal dissemination threshold for the classifier at a

specific time instant, we cannot guarantee that the estimated threshold is good at every time instant throughout the filtering process. One solution is to adjust the dissemination threshold dynamically throughout the filtering process so that the filtering system is more adaptable in an interactive environment.

5.2 Existing Threshold Learning Techniques

In [7], Callan suggested to improve the Exponentiated-Gradient (EG) algorithm [18] by adjusting the threshold of the classifier during the filtering process so that the filtering system can find a good threshold dynamically. The adjusted threshold is set somewhere between the average score of the retrieved relevant documents and the average score of the retrieved non-relevant documents. In order to reduce the risk of setting a poor threshold, the system can start to adjust the threshold after retrieving ten relevant and ten non-relevant documents. One characteristic that Callan mentioned is that the threshold learning technique is not misled by occasional low-scoring relevant document or high-scoring non-relevant document. However, we think that this characteristic holds only when the number of retrieved relevant documents and the number of retrieved non-relevant documents are high enough. Moreover, we observe that the average score of the retrieved rele-

vant documents is not necessarily always greater than that of the retrieved non-relevant documents especially at the beginning of the filtering process. If the average score of the retrieved relevant documents is lower than that of the retrieved non-relevant documents, the threshold learning technique will not work. Also, it is not fully automatic because users need to decide the position of the new threshold within the two average scores in advance.

Zhai et al. [45] also proposed to adjust the dissemination threshold dynamically in their TREC-7 [17] adaptive text filtering track entry. They proposed a technique, namely the beta-gamma adaptive threshold regulation, to adjust the threshold by interpolating between an "optimal" threshold and "zero" threshold. The *optimal threshold* is the threshold that yields the highest performance, given the newly updated feature weight vector, over the accumulated training documents. The *zero threshold* is the highest threshold below the optimal threshold that gives a non-positive performance over the training documents. The threshold learning technique involves two parameters. One limitation of this technique is that we need to set the parameters in advance which may vary for different document corpora.

5.3 A New Threshold Learning Algorithm

We propose to learn a local optimal dissemination threshold with respect to the performance measure dynamically during the filtering process. Our threshold learning algorithm is fully automatic which does not need any user intervention. Step 4(c) in Figure 4.1 of our REPGER algorithm is responsible for learning the dissemination threshold.

We propose to adjust the dissemination threshold based on the historical information. At any time instant during the filtering process, we have the current $Pool_R$ and a collection of retrieved documents. The maximum and the minimum feature weights of the features in $Pool_R$ are first obtained from the features in $Pool_R$. The dissemination threshold will then be determined somewhere between the two values. The idea is to choose the threshold with the best value with respect to the performance measure when using the threshold and the current $Pool_R$ to filter the previously retrieved documents. The reason for using the maximum and the minimum feature weights of the features in $Pool_R$ as the boundaries is that these two boundaries can restrict the new dissemination threshold within a reasonable range for retrieving other incoming documents. The adjusted threshold may be too high or too low if there is no boundary constraint. When there is a range of thresholds giving the best performance, we choose the mid-point of the range. If there

are two or more non-consecutive thresholds giving the best performance, the lowest one will be chosen because it can help the system maintain a higher possibility of retrieving another incoming document. By using this mechanism, a new threshold can be learned whenever the system retrieves an incoming document. In summary, the threshold learning procedure is summarized as follows:

1. Obtain $\max\{w_k\}$ and $\min\{w_k\}$ of the features in the current $Pool_R$ to form a value range [$\min\{w_k\}$, $\max\{w_k\}$].

2. Divide the value range into $b + 1$ intervals to get b boundary points. One of the b boundary points will be the new threshold. (We used $b = 20$ in our current implementation.)

3. Use each boundary point and the current $Pool_R$ to filter the previously retrieved documents. Every boundary point will have a corresponding value of the performance measure after the filtering process.

4. Choose the boundary point with the best performance with respect to the performance measure given by the user as the new threshold. If there are consecutive boundary points giving the best performance, the middle one or the one immediately lower than the mid-point will be chosen. If there are two or more non-consecutive boundary points giving the best performance, the lowest one will be used.

Chapter 6

Empirical Evaluations

6.1 Experimental Methodology

We have conducted experiments similar to the adaptive text filtering track in the Seventh Text REtrieval Conference (TREC-7) [17] in 1998. It was the first time for TREC to organize the adaptive text filtering track. Incoming documents arrive in chronological order. The filtering system starts with a topic description. For each incoming document, it needs to make a binary decision to accept or reject the document. When the system retrieves a document, the filtering profile is learned and updated by using the relevance judgment of that document. Three corpora, namely, AP (Associated Press), FBIS (Foreign Broadcast Information Service) and WSJ (Wall Street Journal) were used. Associated Press (AP) World-stream is an amalgamation

of all the AP-produced international services. The English language copy usually originates in or is of interest to areas outside the United States; the service also produces copy in French, German, Swedish, Dutch and Spanish. The Foreign Broadcast Information Service (FBIS) is a United States government operation which translates the text of daily broadcasts, government statements, and select news stories from non-English sources around the world. The Wall Street Journal (WSJ) is a trusted and reliable business newspaper. Thus, the WSJ corpus is a collection of the articles in this newspaper. We have fully implemented our REPGER algorithm and conducted experiments on these three document corpora to study the effectiveness of our algorithm in adaptive filtering. We also compared the performance of our REPGER algorithm with two on-line learning algorithms, namely, the basic EG algorithm [18] and the improved EG algorithm in [7] (called EG-C in this thesis) on the three document corpora. The EG-C algorithm enhances the EG algorithm by dynamically adjusting the dissemination threshold and the target values.

As we mentioned earlier, the EG-based algorithms integrated with the Infinite Attribute Model [4] can be applied to adaptive filtering problems so that the full set of distinct features need not be known in advance. Nevertheless, in our experiments, we give the EG and the EG-C algorithms an advantage by providing the full set of distinct features in advance. In

contrast, our REPGER algorithm does not have such information. At the beginning of the filtering process, for the EG and the EG-C algorithms, we treat the topic description as a relevant document and update the filtering profile by the weight update formula using the binary feature representation. Each incoming document is represented by a unit vector or a set of features depending on which algorithm is used. Features are composed of words extracted from documents or topic descriptions after stop-word removal and word stemming. For the topic descriptions, words in the description or the concept fields (marked by SGML tags <desc> and <con>) are extracted. Normalized TF-IDF is adopted in the document representation. For each document corpus, the initial IDF statistics were derived from an unrelated reference corpus. This technique is commonly used for obtaining a more accurate estimate for the initial statistics. The reference corpus used for each document corpus in our experiments is shown in the following table.

Document corpus	Reference corpus
AP	WSJ
FBIS	AP
WSJ	AP

During the filtering process, the IDF used for each incoming document is calculated incrementally using the initial statistics and the historical information collected as shown in Equation 3.3.

We follow the set-based utility measure, namely F3, proposed in TREC-7 [17] to evaluate the performance of an adaptive text filtering system. The measure is defined as:

$$\text{F3} = 4A - B \tag{6.1}$$

where A is the number of retrieved relevant documents and B is the number of retrieved non-relevant documents. F3 assigns a value of 4 units to each retrieved relevant document and a cost of 1 unit to each retrieved non-relevant document. The larger the F3 score, the better the filtering system performs for a given topic. Filtering according to a utility measure is equivalent to filtering by estimated probability of relevance. Therefore, filtering according to F3 is equivalent to setting the probability threshold to 0.2, i.e. retrieve the document if the probability of relevance of the document is greater than 0.2 [1].

For each document corpus, we have conducted experiments for different algorithms on a set of topics. We evaluated the performance by considering the macro measures of all topics. In particular, the set F3 measure is derived from the macro figures in Equation 6.1. We used the set F3 measure to reflect how good an algorithm performs over a set of topics.

[1] Readers can find the general formula for converting a utility measure into a probability threshold in [26].

6.2 Experimental Settings

For the AP corpus, we used 164,597 documents from 1988 and 1989. All 50 topics from TREC-7 [17] filtering track (topic number 1-50) were used. For the FBIS corpus, we used 130,471 documents from 1993 and early 1994. We used 38 topics from TREC-5 [42] routing/filtering track in our experiments which have hundreds of relevance judgments for each topic. For the WSJ corpus, we used 98,732 documents from 1987 to 1989. 50 topics from TREC-7 routing track (topic number 51-100) were used.

For each document corpus, we varied the parameters of each algorithm in an attempt to study the performance for a particular algorithm. We also recorded the on-going filtering performance during the filtering process to observe the adaptive behavior of each algorithm. For each document corpus, we evenly divide the full set of documents into five intervals. We then record the filtering performance of each algorithm at the end of each interval. By observing the change of the performance of each algorithm through the five intervals, we observe the adaptive behavior of each algorithm. For the EG algorithm, we varied the dissemination threshold, the target value for relevant documents and the learning rate η. We set the target value for non-relevant documents to 0 because it is reasonable to obtain a zero inner product $\vec{W} \cdot \vec{D}$ for a non-relevant document and a learned classifier in prac-

tice. For the EG-C algorithm, we followed the desirable parameter setting for the value of R [2] in [7] and varied the initial dissemination threshold, the threshold learning starting point and the learned threshold position between the average score of the retrieved relevant documents and the average score of the retrieved non-relevant documents. For our REPGER algorithm, we followed the parameter setting for the value of R in [7]. We set the target values for relevant documents and for non-relevant documents to 0 and 0.5 respectively. We varied the initial dissemination threshold, the value of β and the threshold learning starting point to investigate their effects. In addition, we also conducted experiments with and without the GER to see how the GER contributes to the filtering process. REPGER0 denotes the trial of REPGER without using the GER. Experiment was also conducted to find out whether the assumption of feature characteristic in Section 4.2 is valid or not on the three document corpora. In order to achieve this purpose, we designed two variations of the REPGER algorithm. Both are based on the REPGER framework without using GER and threshold learning mechanism. REPGER1 applies the REP while REPGER2 considers all features that the filtering system has encountered in predicting document relevance. Table 6.1 summarizes the parameter settings for the algorithms.

In order to compare the effectiveness of the algorithms throughout the

[2]Note that the learning rate $\eta = 2/(3R^2)$.

Algorithm	**Parameter Setting**
EG	target value for non-rel. doc. (0), target value for rel. doc. (0.005-1), R (0.8 or KW)
EG-C	target values (MinMax), R (0.8), threshold learning starting point (0-30 retrieved rel. and non-rel. doc. each), learned threshold position (0%, 25%, 50% and 75%)
REPGER	target values (0.5, 0), R (0.8), β (1.3-1.8), threshold learning starting point (3-30 retrieved doc.)
REPGER0	REPGER without using GER
REPGER1	base on the REPGER framework, follow the parameter setting of REPGER, only use REP
REPGER2	base on the REPGER framework, follow the parameter setting of REPGER, consider all features in predicting doc. rel.

Table 6.1: *The parameter settings for the algorithms.*

filtering process, we present the on-going filtering performance of the trials corresponding to the best parameter combination of each algorithm. Besides comparing the performance of the three algorithms, we also compare the performance of our REPGER algorithm on the AP corpus with the participants of the adaptive text filtering track in the latest TREC-7 [17]. Note that this track only conducted filtering runs on the AP corpus.

6.3 Experimental Results

The detailed experimental results of the algorithms on the AP, FBIS and WSJ document corpora were presented in Appendix A, B and C respectively. For each algorithm, we only show the runs which are necessary to give the trend of the filtering performance for each document corpus. From the experimental results, we observe that the EG and the EG-C algorithm are very sensitive to the parameter settings, including the dissemination threshold and the target value for relevant documents. For the EG algorithm, the KW technique suggested by Kivinen and Warmuth in [18] does not give a significant improvement. It is because the KW technique was originally designed for batch learning such as batch filtering. In batch filtering, the learning rate set by the KW technique is fixed throughout the learning process. However, when it is applied to adaptive filtering, the learning rate is varied according

to the constraint of the KW technique. Therefore, it may not be suitable for adaptive filtering. For the EG-C algorithm, it performs better than the EG algorithm on the three document corpora. It shows that the improvements proposed by Callan make the EG algorithm more effective in adaptive filtering. As we discussed in Section 3.2.2, the dissemination threshold is adjusted within two threshold bounds. We observe that the EG-C algorithm always gives the best filtering performance at the position 0%, i.e. the adjusted dissemination threshold is set to the lower bound [3]. The performance may be higher if we set the adjusted threshold lower than the lower bound. It means that the lower bound may be too high.

Table 6.2 summarizes the best performance of REPGER1 and REPGER2 on the three document corpora. The best performance of REPGER1 is always higher than that of REPGER2 on these corpora. It shows that considering only the features which appear in the topic description and the retrieved relevant documents in predicting document relevance can improve the filtering performance. This observation implies that our assumption on the feature characteristic in Section 4.2 is reasonable and the REP concept helps improve the performance in adaptive filtering.

Figures 6.1, 6.2 and 6.3 show the effect of the initial dissemination threshold on the performance of the EG, the EG-C and the REPGER algorithms

[3]The lower bound is the average score of the retrieved non-relevant documents.

Corpus	REPGER1	REPGER2
AP	23	4
FBIS	9	1
WSJ	376	78

Table 6.2: *The best performance, set F3, of REPGER1 and REPGER2 on the three document corpora.*

on the FBIS corpus respectively. We plotted two sub-figures to show the effect of using KW for the EG algorithm in Figure 6.1. Performance of using different target values for relevant documents is also shown in the sub-figures. In Figure 6.2, we plotted the performance of setting different adjusted threshold positions of the EG-C algorithm with threshold learning starting point at 30 [4]. For the REPGER algorithm, we plotted the performance of setting different values of β (beta) with threshold learning starting point at 15 [5] in Figure 6.3. We aim at observing the sensitivity of parameter settings of different algorithms from these figures. We find that our REPGER algorithm gives satisfactory performance over a range of initial dissemination threshold

[4]The EG-C algorithm gives the best performance with threshold learning starting point at 30 on the FBIS corpus.

[5]We randomly choose a value of the threshold learning starting point to show the effect of different values of β (beta) and initial dissemination threshold of the REPGER algorithm on the FBIS corpus.

and a range of β (beta). The performance is less sensitive to the parameters. In contrast, the performance of the EG and the EG-C algorithms is more sensitive to the initial dissemination threshold. For example, the EG and the EG-C algorithms can only give non-negative performance over a narrow range of 0.000094-0.000106 and 0.000092-0.000108 respectively for the initial dissemination threshold parameter. These characteristics are also observed on the other two corpora.

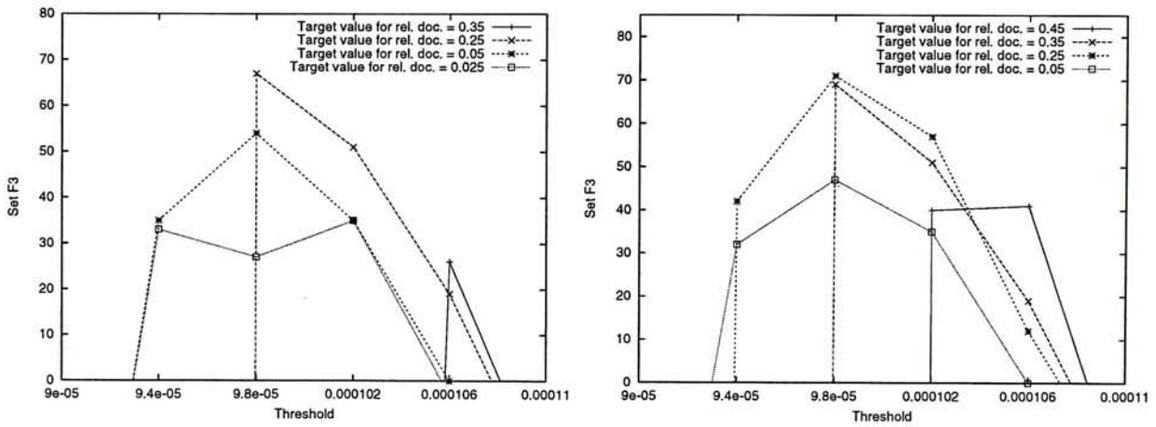

Figure 6.1: *The filtering performance of the EG algorithm with different dissemination threshold and target value for relevant documents settings on the FBIS corpus. Without KW on the left and with KW on the right.*

Figure 6.4 depicts the effects of the parameter β (beta) and the initial dissemination threshold on the performance of our REPGER algorithm on the FBIS corpus. We plotted three initial dissemination thresholds with a range of β for our REPGER algorithm and also plotted the best performance of the EG and the EG-C algorithms for reference. The threshold learning starting

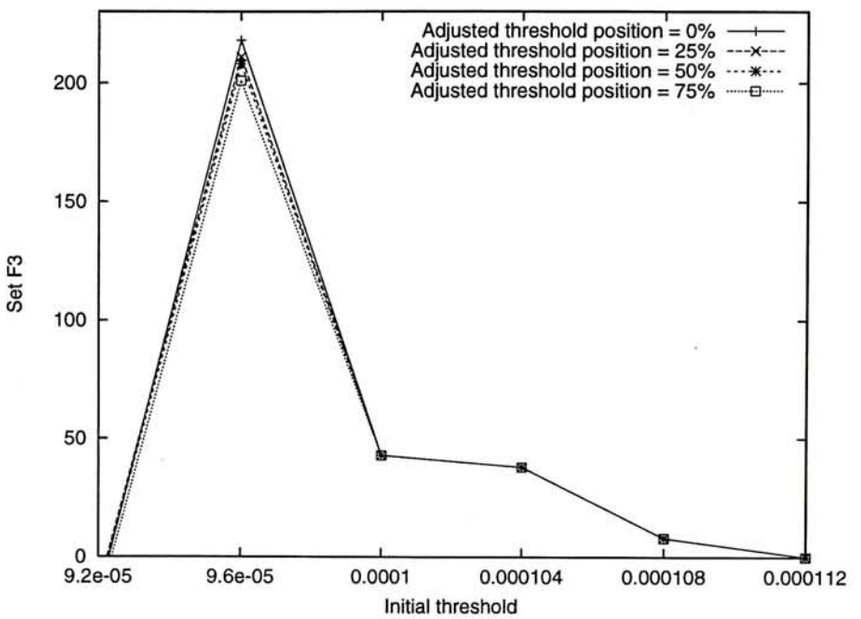

Figure 6.2: *The filtering performance of the EG-C algorithm with different initial dissemination threshold and adjusted threshold position settings at threshold learning starting point = 30 on the FBIS corpus.*

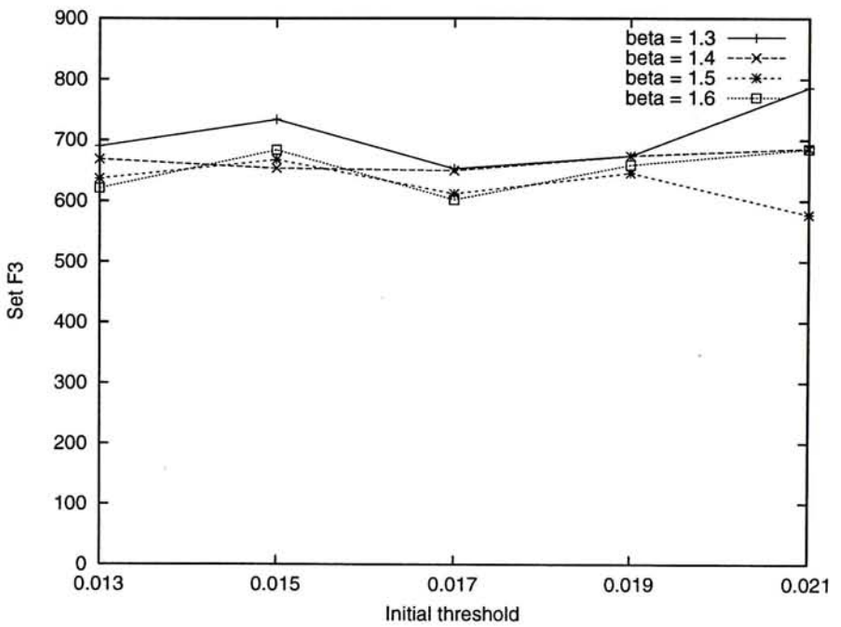

Figure 6.3: *The filtering performance of the REPGER algorithm with different β (beta) and initial dissemination threshold settings at threshold learning starting point = 15 on the FBIS corpus.*

point of our REPGER algorithm was set to 15 [6]. This empirical result shows two characteristics of our REPGER algorithm. First, we observe that our REPGER algorithm performs better than the EG-C algorithm and the EG-C algorithm gives better performance than the EG algorithm. Second, the REPGER algorithm gives stable and high performance on a range of values of β (beta). This shows that the performance of the REPGER algorithm is less sensitive within a range of values of β.

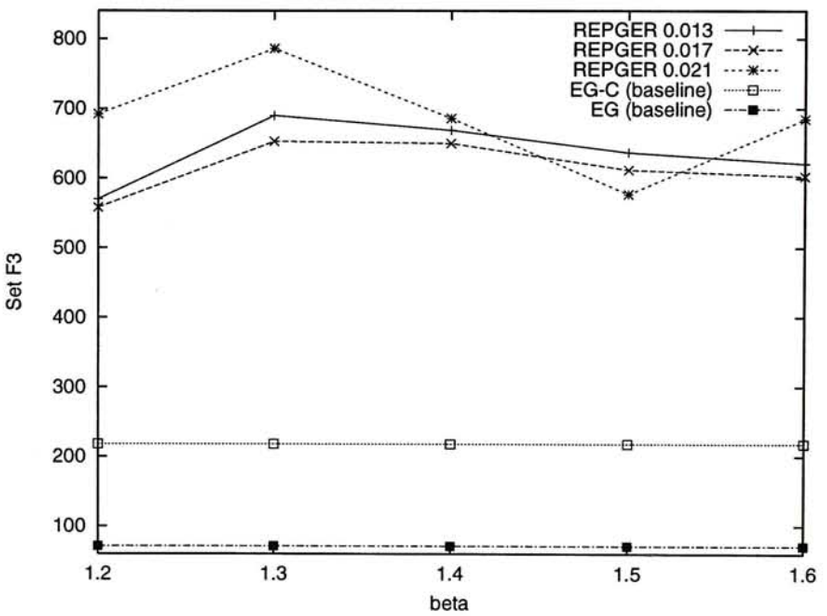

Figure 6.4: *The filtering performance of the REPGER algorithm with different initial dissemination threshold and β (beta) settings at threshold learning starting point = 15 on the FBIS corpus.*

[6] We randomly choose a value of the threshold learning starting point to show the effect of different values of β (beta) and initial dissemination threshold of the REPGER algorithm on the FBIS corpus.

Figure 6.5 shows the effects of the threshold learning starting point parameter and the initial dissemination threshold on the performance of our REPGER algorithm on the FBIS corpus. We plotted three initial dissemination thresholds with a range of threshold learning starting points for our REPGER algorithm and also plotted the best performance of the EG and the EG-C algorithms for reference. The β (beta) of our REPGER algorithm was set to 1.6 [7]. We observe one more characteristic of our REPGER algorithm from this empirical result. It is that our REPGER algorithm gives satisfactory performance over a range of threshold learning starting points and the performance is always better than that of the EG and the EG-C algorithms.

Figures 6.6, 6.7 and 6.8 depict the on-going filtering performance of the best trials of the EG, the EG-C, the REPGER0 and the REPGER algorithms on the three corpora during the whole filtering task. Recall that REPGER0 denotes REPGER without using the GER. They demonstrate that the adaptive learning performance of the REPGER0 and the REPGER algorithms is much higher than that of the others. For the three corpora, we observe that the adaptive learning performance of the EG algorithm is very

[7] We randomly choose a value of β (beta) to show the effect of different values of threshold learning starting point and initial dissemination threshold of the REPGER algorithm on the FBIS corpus.

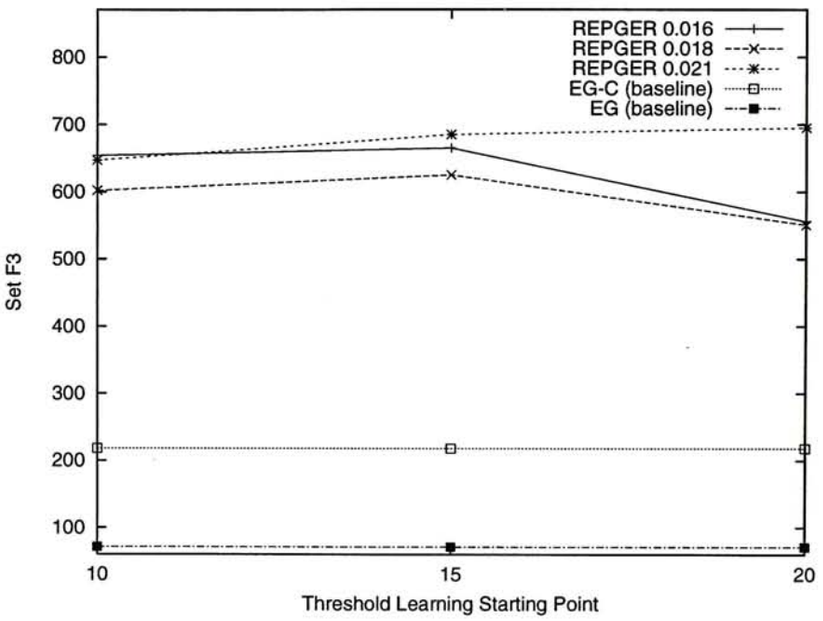

Figure 6.5: *The filtering performance of the REPGER algorithm with different initial dissemination threshold and threshold learning starting point settings at β (beta) = 1.6 on the FBIS corpus.*

low. It may be due to the problem that the EG algorithm does not have a mechanism to maintain a pool of selective features with potentially high predictive power in an interactive environment. For the FBIS corpus, the REPGER algorithm starts with a performance lower than that of the EG and the EG-C algorithms. It seems that the topics in the FBIS corpus is difficult for our algorithm to learn. We found that the initial features extracted from the topic descriptions are not informative. Our algorithm starts with a filtering profile containing only the features extracted from the topic description. This makes our algorithm retrieving some non-relevant documents at the beginning of the filtering process for these topics. Nevertheless, it achieves better performance than the EG-based algorithms after processing some documents. Note that the EG and the EG-C algorithms know the full set of distinct features in advance to construct their filtering profiles.

When concentrating on the performance of the REPGER0 and the REPGER algorithms on the three corpora, we observe that the REPGER algorithm always give a better performance. When the topics are difficult to learn such as the FBIS corpus, it can give a considerable contribution to the filtering system. The GER reduces the performance damage at the beginning stage of the filtering process. In other words, it increase the filtering performance of the algorithm at the beginning stage. This is exactly what the GER intends to achieve. By having a better performance at the beginning stage, the per-

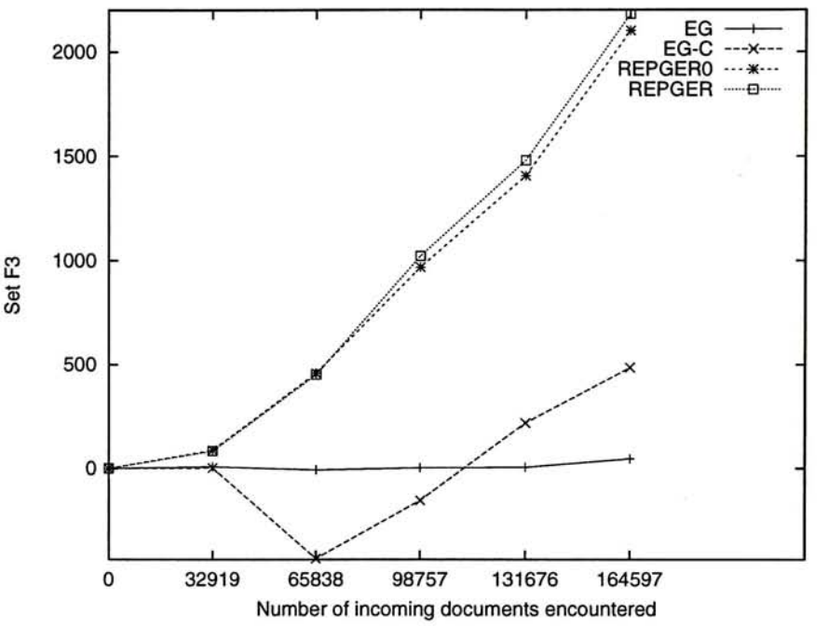

Figure 6.6: *The on-going filtering performance of the four algorithms chosen by the best parameter combinations on the AP corpus.*

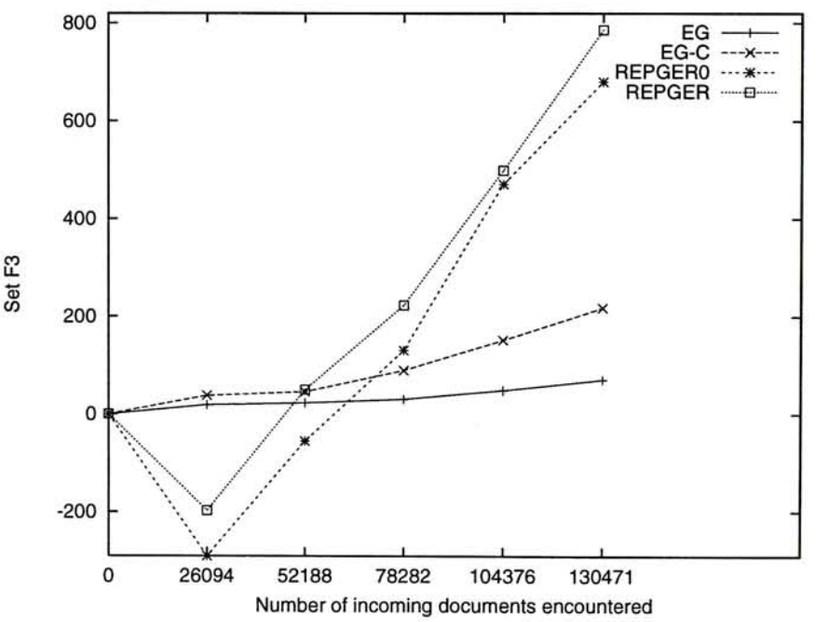

Figure 6.7: *The on-going filtering performance of the four algorithms chosen by the best parameter combinations on the FBIS corpus.*

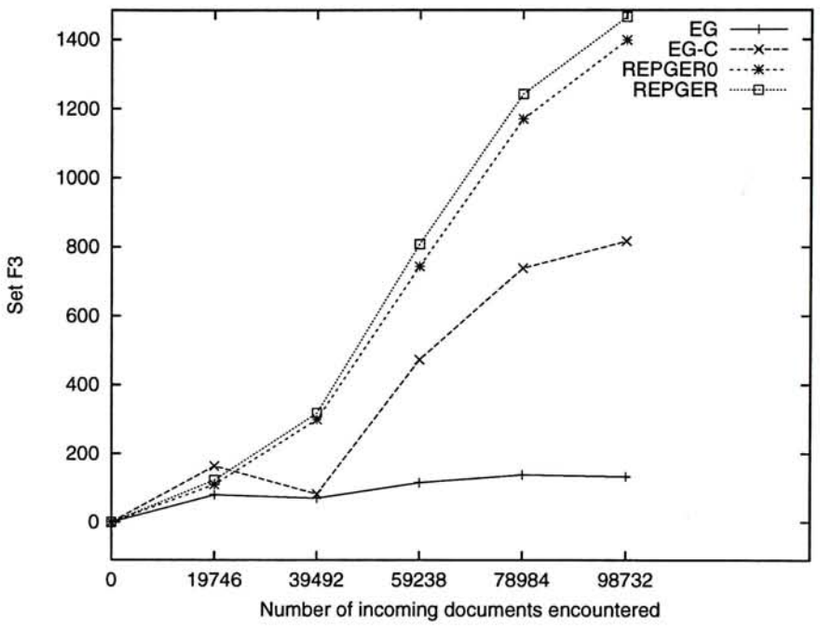

Figure 6.8: *The on-going filtering performance of the four algorithms chosen by the best parameter combinations on the WSJ corpus.*

formance will keep better than REPGER0 throughout the filtering process. For example, on the FBIS corpus, the percentage increase of the performance at the end of the filtering process is 15.59% [8] which is a significant improvement. This validates the use of the GER for tackling the problem of lack of training examples for the information filtering system in adaptive filtering.

Table 6.3 shows the average set F3 values of the on-going performance of the EG, the EG-C and the REPGER algorithms. For instance, the average set F3 values of the AP corpus is the mean of the five set F3 values shown in Figure 6.6 for the corresponding algorithm. It shows that our REPGER

[8]REPGER gives 786 and REPGER0 gives 680. The percentage increase is (786-680)/680 = 15.59%.

algorithm has higher average utility values demonstrating its effectiveness.

Corpus	EG	EG-C	REPGER
AP	10	22.4	1,042.8
FBIS	38.6	108.8	272.2
WSJ	107.2	454.6	791.2

Table 6.3: *The average set F3 values of the on-going performance of the EG, the EG-C and the REPGER algorithms.*

In comparing the performance with the participants of the adaptive text filtering track in TREC-7 [17], only AP years 88 and 89 could be used since we could not obtain the relevance judgments for the year 90. Similar to most TREC-7 participants, we adopted a topic-by-topic comparison fashion and concentrated on the comparison with their median performance [9]. Comparing with the median performance of the TREC-7 participants can reflect how well an algorithm performs among them. Table 6.4 summarizes the comparison of the performance of our REPGER algorithm with the TREC-7 participants over 50 topics. It depicts the number of topics in which the F3 of our REPGER algorithm is less than, equal to, or greater than the median of the TREC-7 participants. It shows that our REPGER algorithm is

[9] Actually, we can only obtain the minimum, median and maximum performance of the participants for the 50 topics.

comparable to them. Note that no single participant obtained performance consistently better than the median over all topics. Moreover, our algorithm achieves the same performance as their best performance in 5 topics and even achieves better performance than their best performance in 3 topics in the AP year 89 corpus.

Corpus	< median	= median	> median
AP year 88	24	2	24
AP year 89	21	5	24

Table 6.4: *The number of topics in which the F3 of our REPGER algorithm is less than, equal to, or greater than the median of the TREC-7 adaptive text filtering track participants (total 50 topics).*

Chapter 7

Integrating With Feature Clustering

In the vector space model, both the incoming documents and the filtering profile are represented as a feature vector as described in Equation 3.1 and 3.4. Consider now a situation in which n distinct features are available to characterize document content. Each of the n features, f_k, can then be identified with a feature vector $\vec{f_k}$. Hence the incoming document D can be rewritten as

$$\vec{D} = \sum_{i=1}^{n} x_i \, \vec{f_i} \tag{7.1}$$

where x_i, the document weight, is interpreted as the components of the doc-

ument D along the vector $\vec{f_i}$.

In a vector space, a common similarity measure used to compute the similarity between vectors $\vec{a}$ and $\vec{b}$ is the cosine similarity. It is their inner product, $\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos \delta$, where $|\vec{a}|$, $|\vec{b}|$ are the lengths of $\vec{a}$, $\vec{b}$, respectively and δ is the angle between the two vectors. Hence, given a document $\vec{D}$ and a filtering profile $\vec{W}$ represented in the form of Equation 7.1, the profile-document similarity can be computed as

$$\vec{D} \cdot \vec{W} = \sum_{i,j=1}^{n} x_i w_j \, \vec{f_i} \cdot \vec{f_j} \tag{7.2}$$

Computing the similarity value thus depends on a specification of the document and filtering profile components, as well as knowledge of the feature correlations $\vec{f_i} \cdot \vec{f_j}$ for all feature pairs. The feature correlations are not usually available a priori especially in the adaptive filtering setting. In practice the feature-correlation problem is often solved by assuming that the features are in fact uncorrelated, in which case the feature vectors are orthogonal (i.e., $\vec{f_i} \cdot \vec{f_j} = 0$, except when $i = j$ and $\vec{f_i} \cdot \vec{f_i} = 1$). When the n feature vectors are orthogonal, linear independence follows automatically. Assuming that the features are uncorrelated, the profile-document similarity computation of Equation 7.2 is reduced to a simple sum-of-products form of Equation 7.3:

$$\text{sim}(\vec{D}, \vec{W}) = \sum_{i=1}^{n} x_i w_i \tag{7.3}$$

The profile-document similarity measure used in our REPGER algorithm (Step 2 in Figure 4.1) is a variant of the Equation 7.3. It also works under the assumption of feature uncorrelation. However, in fact, this assumption does not hold. Many features are correlated to each other in a document corpus. For example, the words "rear", "steering" and "tire" are correlated.

Feature clustering methods can solve this problem by joining similar features, i.e. features that are correlated, into groups. After feature clustering, the set of distinct features considered in the vector space becomes a set of clusters by joining the similar features into groups. Hence, the assumption that the vectors are orthogonal becomes reasonable.

There is another benefit of feature clustering. It can reduce the feature dimensionality in the vector space. The size of the filtering model can be reduced because the large set of features in $Pool_R$ is reduced to a relatively small set of feature clusters. Hence, the efficiency of calculating the profile-document similarity can be improved.

In this thesis, we investigate the contribution of feature clustering in adaptive filtering. We modify the distributional clustering algorithm proposed in [1] so that the algorithm can be applied in the adaptive filtering setting.

7.1 Distributional Clustering Algorithm

Distributional clustering [33] is an information-theoretic approach that has shown good performance in language modeling. It joins features that induce similar probability distributions among the target concept that co-occur with the features in question. The reasoning behind this is as follows. If two different features "vote" similarly among the possible answer in the task at hand, then the features are correlated and can be joined into a cluster. Baker and McCallum [1] modified the distributional clustering algorithm for text classification. They show that distributional clustering is better than other existing clustering techniques, such as class-based clustering using mutual information [5] and clustering by Latent Semantic Indexing [11], in both reducing feature dimensionality and improving accuracy. Therefore, we choose distributional clustering to integrate with the REPGER algorithm. We modify the algorithm in [1] so that we can apply distributional clustering of features in adaptive filtering.

Consider a random variable over classes, C, and its distribution given a particular feature, f_i. The distribution can be written as $P(C|f_i)$. Note that, in information filtering, the class variable C contains only two classes, *relevant* and *non-relevant*. When features f_i and f_j are clustered together, the new distribution can be estimated as the weighted average of the individual

distributions

$$P(C|f_i \vee f_j) = \frac{P(f_i)}{P(f_i)+P(f_j)}P(C|f_i) + \frac{P(f_j)}{P(f_i)+P(f_j)}P(C|f_j) \quad (7.4)$$

In the context of information filtering, the target variable for the task at hand is the class label. Distributional clustering thus measures the similarity between two features f_i and f_j as the similarity between the class variable distributions they induce: $P(C|f_i)$ and $P(C|f_j)$. Kullback-Leibler (called KL in this thesis) divergence is an information-theoretic score that measures the difference between two probability distributions. The KL divergence between the class distributions induced by f_i and f_j is defined as

$$D(P(C|f_i)\|P(C|f_j)) = \sum_{k=1}^{|C|} P(c_k|f_i)\log(\frac{P(c_k|f_i)}{P(c_k|f_j)}) \quad (7.5)$$

The KL divergence has some odd properties. It is asymmetric, and is infinite when an event with non-zero probability in the first distribution has zero probability in the second distribution. Thus, in Distributional Clustering we use a related measure that does not have these problems. It is the average of the KL divergence of each distribution to their mean distribution, called "KL divergence to the mean". In [1], they use a weighted average instead of a simple average.

$$P(f_i)D(P(C|f_i)||P(C|f_i \vee f_j)) + P(f_j)D(P(C|f_j)||P(C|f_i \vee f_j)) \quad (7.6)$$

In our investigation, we follow the weighted average of the "KL divergence to the mean" in Equation 7.6 to measure the difference between two probability distributions over the class variable.

Now we address the question of how to use the similarity metric to form clusters. We present the clustering algorithm first and then describe how to integrate it with our REPGER algorithm in Section 7.2. We create clusters with deterministic feature membership using a simple greedy agglomerative approach that works well in practice. The number of clusters desired, M, should be specified in advance. At all stages, the algorithm has not more than M clusters. The clusters are initialized with M features that have highest average mutual information with the class variable. Mutual information is a criterion commonly used in statistical language modeling of word associations and related applications [9, 15]. The mutual information between the feature f_k and the class label c_i is defined as

$$I(f_k, c_i) = \log(\frac{P(f_k \wedge c_i)}{P(f_k)P(c_i)}) \quad (7.7)$$

and the average mutual information is

$$I_{avg}(f_k) = \sum_{i=1}^{|C|} P(c_i) I(f_k, c_i) \tag{7.8}$$

After forming the first M clusters, the remaining features are joined to one of the M clusters until each feature belongs to one of the clusters. Figure 7.1 summarizes the clustering algorithm.

1. Sort the features by average mutual information with the class variable.

2. Initialize the M clusters as singletons with the top-ranked M features.

3. Loop until all features have been put into one of the M clusters:

   - Group the next feature in the sorted list into one of the M clusters that are most similar to the feature according to Equation 7.6.

Figure 7.1: *The Clustering Algorithm.*

7.2 Integrating With Our REPGER Algorithm

We apply the distributional clustering algorithm to the features in $Pool_R$ of our REPGER algorithm. The features with similar probability distributions over the class variable in $Pool_R$ can be joined together into a cluster. The parameters of the cluster such as the *document weight*, x_k, and the *feature weight*, w_k, become the simple average of the parameters of its constituent

features. Since the clustering algorithm is based on the probability distributions of the features, the performance of our REPGER algorithm after starting feature clustering is heavily dependent on the accuracy of the features' probability distributions. We should start the clustering algorithm after the system has retrieved a large number of documents so that the system can maintain accurate estimates on the probability distributions over the class variable. After starting the clustering algorithm, the clusters will be updated every time when the system retrieves an incoming document. The number of clusters M can be different in each update. In adaptive filtering, we cannot estimate the number of features in $Pool_R$ at any time instance in the filtering process. Moreover, this number also varies from different topics. We propose to set the value of M dynamically in the filtering process. We can relate the number of clusters M with the number of features in $Pool_R$ by the following equation:

$$M = ratio_c \, |Pool_R| \tag{7.9}$$

where $ratio_c$ is the ratio used to determine the size of M. After starting feature clustering, the weight update formula and the threshold learning algorithm of our REPGER algorithm also works on the clusters in $Pool_R$.

7.3 Empirical Evaluation

In order to investigate the effect of the clustering algorithm purely in the profile-document similarity (Step 2 in Figure 4.1), we integrate the clustering algorithm with our REPGER algorithm without using the GER. Based on the experimental results in Section 6.3, we chose one topic with the highest number of retrieved documents from each document corpus for our clustering experiment. Table 7.1 shows the details of the topics chosen.

Corpus	Topic number	Number of retrieved documents
AP	22	1,650
FBIS	111	1,810
WSJ	56	431

Table 7.1: *The details of the topics chosen for the clustering experiment.*

We follow the parameter settings of our REPGER algorithm that give the results shown in Table 7.1 and explore a range of parameter values, i.e. the clustering starting point and the value of $ratio_c$, for the clustering algorithm. The clustering algorithm will start if the number of retrieved documents is larger than the clustering starting point. The parameter $ratio_c$ is used to decide the number of clusters, M, as described in Equation 7.9. We concentrate on the difference in performance with and without the clustering

algorithm. Therefore, only the performance after the number of retrieved documents exceeds the clustering starting point will be analyzed.

Corpus	Topic no.	No. of retrieved documents	Clustering starting point	$ratio_c$	F3 without clustering	F3 with clustering
AP	22	1,650	1,600	0.025-0.2	16	33
FBIS	111	1,810	1,800	0.2-0.7	-4	4
WSJ	56	431	400	0.4-0.9	15	13

Table 7.2: *The result of the clustering experiment.*

Table 7.2 summarizes the experimental result. It shows the best performance and the parameter setting that gives the best performance of the algorithm integrating with the feature clustering technique. The clustering starting points for the AP, the FBIS and the WSJ corpora are 1,600, 1,800 and 400, respectively. For the AP corpus, the clustering algorithm helps our REPGER algorithm without the GER attain a performance improvement from 16 to 33 in the $ratio_c$ range 0.025-0.2. It also gives improvement from -4 to 4 in the $ratio_c$ range 0.2-0.7 for the FBIS corpus. However, the performance reduces from 15 to 13 in the $ratio_c$ range 0.4-0.9 for the WSJ corpus. The clustering algorithm gives improvement in the topics of the AP and the FBIS corpora but not in that of the WSJ corpus. It shows that the clustering starting point does affect the performance of the clustering algorithm. The

clustering algorithm does well only when there are enough statistics to estimate the probability distributions over the class variable of the features in $Pool_R$. In addition, the clustering performance may depend on the content of the corpus. If features of a document corpus are less correlated, the clustering algorithm will not do well because it may over-cluster the features. The FBIS corpus may be this type of corpus so that the performance is decreased after applying the clustering technique. In conclusion, this empirical result also shows that clustering correlated features together improves the performance of our REPGER algorithm. This is because, by doing so the assumption of feature uncorrelation in calculating the profile-document similarity becomes more reasonable. However, work needs to be done to investigate how to solve the shortcoming of the deficiency of statistics available in adaptive filtering.

Besides improving the filtering performance, feature clustering can also improve the efficiency of calculating the profile-document similarity since the number of features needed to be considered in calculating the similarity is decreased by clustering correlated features together. However, in adaptive filtering, the computational cost in clustering the features is higher than the time saving in calculating the profile-document similarity. It is because we need to re-cluster the features every time when an incoming document is retrieved. Therefore, much work needs to be done to explore and investigate the feasibility of applying feature clustering in adaptive filtering in the future.

Chapter 8

Conclusions

8.1 Summary

In this thesis, we propose a new on-line learning algorithm, namely the **REPGER (RElevant feature Pool with Good training Example retrieval Rule)** algorithm, for adaptive text filtering problems. Our approach maintains a pool of selective features with potentially high predictive power. Besides using the predicted relevant documents to update its filtering profile, it also retrieves documents which are considered as good training examples. Dissemination threshold can be learned dynamically to maintain a good filtering performance throughout the filtering process. We have conducted experiments on three document corpora, namely, AP, FBIS and WSJ to compare the performance of our REPGER algorithm with two existing

EG-based algorithms [7, 18]. The results demonstrate that our REPGER algorithm offers a more effective filtering performance than the others most of the time. Comparison with the TREC-7 [17] adaptive text filtering track participants was also done. The result shows that our REPGER algorithm is comparable to them. We also investigate the feasibility of applying a feature clustering technique, known as the Distributional Clustering [33], in adaptive filtering. The experimental result shows that the effect of the clustering technique is quite promising.

8.2 Future Work

There are some interesting directions for future work concerning the REPGER algorithm. One area is to improve the learning capability of the REPGER algorithm by considering more informative statistics throughout the filtering process. Another direction is to make the GER more adaptive in the interactive environment of adaptive filtering. Both these directions for future work are discussed below.

First, the effectiveness of the feature weight allocation component of our REPGER algorithm could be improved in an interactive environment. We would use the weight update formula of the EG-based algorithms [7, 18] in our current implementation. However, this would not account for all the

informative statistics of the features such as the number of relevant/non-relevant documents that a particular feature appears. By using these statistics, P(relevant|f_k) and P(non-relevant|f_k) could be calculated incrementally which could help the feature weight update formula determine the magnitude of the change more accurately [1]. For example, suppose a retrieved relevant document contained two features f_i and f_j with the same document weight (i.e., $x_i = x_j$), the magnitudes of the feature weight updates of the two features would be the same by using Equation 3.6. However, if P(relevant|f_i) was larger than P(relevant|f_j) and P(non-relevant|f_i) was smaller than P(non-relevant|f_j), the feature weight updates of the two features should not be the same. The magnitude of the feature weight update of the feature f_i should be larger than that of the feature f_j.

Second, we also intend to study in more detail the behavior of the Good training Example retrieval Rule (GER); in particular, how its effectiveness varies with different topics. The relative importance of the new features and the features in $Pool_R$ considered by the GER may vary from different topics. We believe that a self-learning mechanism to adjust the relative importance for different topics will help improve the overall effectiveness of the filtering system.

[1] P(relevant/non-relevant|f_k) is the probability of relevance/non-relevant of a document, which contains the feature f_k.

On a different area, the filtering profile initialization mechanism in REPGER was not fully exploited due to time constraints. We have only used the simplest method for filtering profile initialization just for showing the effectiveness of our REPGER algorithm. The filtering performance of our REPGER algorithm, however, could be improved by carrying out query expansion in the filtering profile initialization process. Query expansion is a process to expand the set of features extracted from the original topic description in order to have a more accurate filtering profile before starting the filtering process. For example, one could introduce synonyms to the filtering profile. This technique would be particularly useful when the topic description is not informative.

— END —

Bibliography

[1] L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 96–103, 1998.

[2] M. Balabanović and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[3] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[4] A. Blum. Learning Boolean functions in an infinite attribute space. *Machine Learning*, 9:373–386, 1992.

[5] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pieta, and J. C. Lai. Class-based *n*-gram models of natural language. *Computational Lingusitics*, 18(4):467–479, 1992.

[6] J. Callan. Document filtering with inference networks. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269, 1996.

[7] J. Callan. Learning while filtering documents. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 224–231, 1998.

[8] J. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[9] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. In *Proceedings of ACL 27*, pages 76–83, Vancouver, Canada, 1989.

[10] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–315, 1996.

[11] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[12] D. Eichmann, M. Ruiz, and P. Srinivasan. Cluster-based adaptive and batch filtering. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, U.S.A., 1998.

[13] D. A. Evans, K. Ginther-Webster, M. Hart, R. G. Lefferts, and I. A. Monarch. Automatic indexing using selective NLP and First-Order thesauri. In *Intelligent Text and Image Handling. Proceedings of a Conference, RIAO'91*, pages 624–644, 1991.

[14] D. A. Evans and R. G. Lefferts. CLARIT-TREC experiments. *Information Processing and Management*, 31(3):385–395, 1995.

[15] R. Fano. *Transmission of Information.* MIT Press, Cambridge, MA, 1961.

[16] D. A. Hull. The TREC-6 filtering track: Description and analysis. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, U.S.A., 1997.

[17] D. A. Hull. The TREC-7 filtering track: Description and analysis. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, U.S.A., 1998.

[18] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Technical report, UCSC-CRL-94-16, Basking Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, CA, 1994.

[19] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[20] K. L. Kwok. Experiments with a component theory of probabilistic information retrieval based on single terms as document components. *ACM Transactions on Office Information Systems*, 8:363–386, 1990.

[21] K. L. Kwok. A network approach to probabilistic information retrieval. *ACM Transactions on Office Information Systems*, 13(3):324–353, 1995.

[22] K. L. Kwok, L. Grunfeld, M. Chan, N. Dinstl, and C. Cool. TREC-7 ad-hoc, high precision and filtering experiments using PIRCS. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, U.S.A., 1998.

[23] W. Lam and C. Y. Ho. Using a generalized instance set for automatic text categorization. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 81–89, 1998.

[24] W. Lam and K. L. Yu. An intelligent adaptive filtering agent based on an on-line learning model. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999. (poster paper), To appear.

[25] K. Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of the Fourteenth International Conference, Machine Learning*, pages 331–339, 1995.

[26] D. D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–254, 1995.

[27] D. D. Lewis. The TREC-5 filtering track. In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 75–96, U.S.A., 1996.

[28] M. E. Maron and L. J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of ACM*, 7:216–244, 1960.

[29] N. Milic-Frayling, C. X. Zhai, X Tong, P. Jansen, and D. A. Evans. Experiments in query optimization, the CLARIT system TREC-6 report. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 415–454, U.S.A., 1997.

[30] J. Mostafa, S. Mukhopadhyay, W. Lam, and M. Palakal. A multilevel

approach to intelligent information filtering: Model, system, and evaluation. *ACM Transactions on Information Systems*, 15(4):368–399, 1997.

[31] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[32] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert : Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence*, pages 54–61, Portland, OR, 1996.

[33] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, 1993.

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

[35] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Computer Supported Cooperative Work Conference*, New York, 1994.

[36] S. E. Robertson and J. K. Sparck. Relevance weighting of search terms. *Journal of American Society for Information Science*, 27:129–146, 1976.

[37] J. J. Rocchio. *The SMART Retrieval System - Experiments in Automatic Document Processing*, chapter Relevance feedback in information retrieval, pages 313–323. Prentice Hall, 1971.

[38] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGill-Hill, 1983.

[39] B. Sheth. A learning approach to personalized information filtering. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1994.

[40] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*, 1993.

[41] H. R. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[42] E. M. Voorhees and D. Harman. Overview of the Fifth Text REtrieval Conference (TREC-5). In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 1–28, U.S.A., 1996.

[43] Y. W. Yan and H. Garcia-Molina. SIFT - A tool for wide-area information dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, pages 177–186, 1995.

[44] K. L. Yu and W. Lam. A new on-line learning algorithm for adaptive text filtering. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 156–160, 1998.

[45] C. X. Zhai, P. Jansen, E. Stoica, N. Crot, and D. A. Evans. Threshold calibration in CLARIT adaptive filtering. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, U.S.A., 1998.

Appendix A

Experimental Results On The AP Corpus

A.1 The EG Algorithm

	Target value for relevant documents				
Threshold	0.35	0.25	0.15	0.05	0.025
0.000078	—	-13,130	-3,377	-2	0
0.000082	-17,384	41	13	13	13
0.000086	-699	11	11	12	12
0.00009	0	—	—	—	—

Table A.1: *The set F3 of the EG algorithm using KW on the AP corpus.*

	Target value for relevant documents				
Threshold	0.125	0.1	0.075	0.05	0.025
0.000074	-10,403	-5,146	-1,442	-131	-89
0.000078	-7,411	45	11	-4	-1
0.000082	15	13	13	13	13
0.000086	11	11	11	12	12

Table A.2: *The set F3 of the EG algorithm on the AP corpus.*

A.2 The EG-C Algorithm

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.000066	-289	-549	-703	-1,004
0.00007	139	-64	-241	-334
0.000074	143	97	66	-109
0.000078	3	-2	0	1
0.000082	18	13	14	16
0.000086	11	11	11	11

Table A.3: *The set F3 of the EG-C algorithm with threshold learning starting point at 0 on the AP corpus.*

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.000066	-732	-693	-740	-699
0.00007	484	376	119	101
0.000074	276	260	203	152
0.000078	97	85	85	63
0.000082	80	76	72	73
0.000086	22	22	24	25

Table A.4: *The set F3 of the EG-C algorithm with threshold learning starting point at 5 on the AP corpus.*

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.000066	-1,235	-1,019	-1,047	-1,124
0.00007	77	120	25	-93
0.000072	247	257	236	130
0.000074	256	233	187	141
0.000076	431	344	271	254
0.000078	159	147	103	83
0.000082	111	109	110	96
0.000086	-38	-35	-30	-19

Table A.5: *The set F3 of the EG-C algorithm with threshold learning starting point at 10 on the AP corpus.*

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.00007	-1,132	-511	-669	-887
0.000074	120	103	21	-25
0.000078	88	73	71	75
0.000082	-58	-38	-34	-25
0.000086	-119	-99	-82	-74

Table A.6: *The set F3 of the EG-C algorithm with threshold learning starting point at 15 on the AP corpus.*

A.3 The REPGER Algorithm

	Value of β (beta)			
Initial threshold	1.3	1.4	1.5	1.6
0.001	1,190	1,958	1,988	1,948
0.002	1,462	1,859	1,936	1,891
0.003	1,742	2,209	2,043	1,992
0.004	1,783	2,047	2,001	1,963
0.005	1,580	2,046	2,025	1,976
0.006	1,832	2,013	1,984	1,935
0.007	948	1,138	1,124	1,063
0.008	1,032	1,141	1,116	1,065
0.009	1,861	1,932	2,054	2,003
0.01	2,025	2,061	2,245	2,251
0.011	1,934	2,038	2,397	2,401
0.012	1,184	1,294	1,650	1,656

Table A.7: *The set F3 of the REPGER algorithm with threshold learning starting point at 3 on the AP corpus.*

	Value of β (beta)			
Initial threshold	1.3	1.4	1.5	1.6
0.001	1,402	1,673	1,614	1,595
0.002	1,456	1,922	1,829	1,829
0.003	1,475	2,024	1,941	1,941
0.004	1,894	2,088	2,032	2,032
0.005	1,725	1,935	1,918	1,867
0.006	1,647	2,146	2,111	2,066
0.007	1,302	1,903	1,848	1,825
0.008	1,681	2,181	2,159	2,101
0.009	1,651	1,995	2,121	2,067
0.01	1,570	1,828	2,020	2,001
0.011	1,671	1,954	2,147	2,128
0.012	995	1,280	1,452	1,433

Table A.8: *The set F3 of the REPGER algorithm with threshold learning starting point at 5 on the AP corpus.*

	Value of β (beta)			
Initial threshold	1.3	1.4	1.5	1.6
0.001	1,625	1,574	1,574	1,574
0.002	1,556	1,686	1,660	1,660
0.003	1,416	1,552	1,526	1,526
0.004	1,705	1,656	1,605	1,605
0.005	1,825	1,804	1,738	1,738
0.006	1,464	1,755	1,699	1,699
0.007	1,786	1,753	1,702	1,702
0.008	1,507	1,808	1,747	1,747
0.009	948	1,127	1,059	1,059
0.01	889	1,215	1,080	1,045
0.011	683	965	830	795
0.012	1,418	1,410	1,275	1,222

Table A.9: *The set F3 of the REPGER algorithm with threshold learning starting point at 10 on the AP corpus.*

Appendix B

Experimental Results On The FBIS Corpus

B.1 The EG Algorithm

	Target value for relevant documents			
Threshold	0.35	0.25	0.05	0.025
0.00009	—	—	-98	-95
0.000094	—	-10,797	35	33
0.000098	-7,652	67	54	27
0.000102	-574	51	35	35
0.000106	26	19	0	-3
0.00011	-23	-24	—	—

Table B.1: *The set F3 of the EG algorithm on the FBIS corpus.*

	Target value for relevant documents			
Threshold	0.45	0.35	0.25	0.05
0.00009	—	-12,922	-2,301	-97
0.000094	-14,204	-9,401	42	32
0.000098	-11,460	69	71	47
0.000102	40	51	57	35
0.000106	41	19	12	0
0.00011	-26	-24	-24	-24

Table B.2: *The set F3 of the EG algorithm using KW on the FBIS corpus.*

B.2 The EG-C Algorithm

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.000088	-255	-249	-243	-243
0.000092	16	23	26	-6
0.000096	86	74	75	76
0.0001	43	43	43	43
0.000104	38	38	38	38
0.000108	8	8	8	8
0.000112	0	0	0	0

Table B.3: *The set F3 of the EG-C algorithm with threshold learning starting point at 10 on the FBIS corpus.*

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.000092	-18	-14	-16	-21
0.000096	218	211	208	201
0.0001	43	43	43	43
0.000104	38	38	38	38
0.000108	8	8	8	8
0.000112	0	0	0	0

Table B.4: *The set F3 of the EG-C algorithm with threshold learning starting point at 30 on the FBIS corpus.*

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.000092	-103	-102	-94	-90
0.000096	70	114	102	73
0.0001	43	43	43	43
0.000104	38	38	38	38
0.000108	8	8	8	8
0.000112	0	0	0	0

Table B.5: *The set F3 of the EG-C algorithm with threshold learning starting point at 50 on the FBIS corpus.*

B.3 The REPGER Algorithm

	Value of β (beta)			
Initial threshold	1.3	1.4	1.5	1.6
0.015	123	88	97	816
0.016	22	614	634	654
0.017	657	620	571	550
0.018	643	666	634	602
0.019	664	629	608	624
0.02	702	90	69	629
0.021	714	106	96	647
0.022	702	168	146	82

Table B.6: *The set F3 of the REPGER algorithm with threshold learning starting point at 10 on the FBIS corpus.*

	Value of β (beta)			
Initial threshold	1.3	1.4	1.5	1.6
0.013	690	669	637	621
0.014	647	678	671	671
0.015	734	654	668	684
0.016	596	588	636	665
0.017	653	650	612	603
0.018	637	680	640	625
0.019	674	674	646	659
0.02	787	110	86	760
0.021	786	686	577	685

Table B.7: *The set F3 of the REPGER algorithm with threshold learning starting point at 15 on the FBIS corpus.*

	Value of β (beta)			
Initial threshold	1.3	1.4	1.5	1.6
0.013	465	510	527	544
0.014	532	538	524	562
0.015	484	469	459	494
0.016	522	516	597	556
0.017	529	573	552	509
0.018	538	571	581	551
0.019	610	601	630	615
0.02	617	426	494	637
0.021	657	634	653	695
0.022	699	645	641	-29
0.023	490	628	634	495
0.024	503	676	690	689
0.025	629	707	728	715

Table B.8: *The set F3 of the REPGER algorithm with threshold learning starting point at 20 on the FBIS corpus.*

Appendix C

Experimental Results On The WSJ Corpus

C.1 The EG Algorithm

	Target value for relevant documents				
Threshold	0.1	0.05	0.025	0.01	0.005
0.000102	—	-784	11	-38	-34
0.00011	—	-76	48	47	44
0.000114	—	-55	78	127	108
0.000122	—	39	102	116	102
0.00013	-182	108	103	91	94
0.000138	26	86	66	55	55
0.000146	88	47	29	29	29
0.000154	18	—	—	—	—

Table C.1: *The set F3 of the EG algorithm on the WSJ corpus.*

	Target value for relevant documents				
Threshold	0.1	0.05	0.025	0.01	0.005
0.000102	-7,885	-54	-29	-32	-26
0.00011	-339	6	44	46	49
0.000114	-238	47	112	133	103
0.000122	-87	91	116	113	102
0.00013	67	110	95	91	92
0.000138	95	72	61	55	55
0.000146	68	32	32	29	29

Table C.2: *The set F3 of the EG algorithm using KW on the WSJ corpus.*

C.2 The EG-C Algorithm

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.00009	-164	-396	-470	-607
0.000098	707	414	453	440
0.000106	600	629	535	124
0.000114	293	314	208	161

Table C.3: *The set F3 of the EG-C algorithm with threshold learning starting point at 0 on the WSJ corpus.*

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.00009	-1,416	-1,043	-1,206	-1,500
0.000098	210	295	224	42
0.000106	763	817	764	746
0.000114	455	456	428	452
0.000122	7	210	188	111
0.00013	154	148	113	101

Table C.4: *The set F3 of the EG-C algorithm with threshold learning starting point at 5 on the WSJ corpus.*

	Adjusted threshold position			
Initial threshold	0%	25%	50%	75%
0.000098	-1,258	-1,001	-1,001	-1,132
0.000102	137	122	-183	-38
0.000106	193	282	406	371
0.00011	126	156	139	108
0.000114	79	144	189	127
0.000118	80	99	182	186
0.000122	68	115	-3	1
0.000126	27	66	56	86

Table C.5: *The set F3 of the EG-C algorithm with threshold learning starting point at 10 on the WSJ corpus.*

C.3 The REPGER Algorithm

	Value of β (beta)		
Initial threshold	1.4	1.5	1.6
0.011	1,616	1,618	1,623
0.013	1,097	1,302	1,306
0.015	1,339	1,629	1,556
0.017	938	1,567	1,488
0.019	901	1,187	1,245
0.021	1,081	1,075	997
0.023	1,144	1,599	1,592
0.025	1,081	1,312	1,297

Table C.6: *The set F3 of the REPGER algorithm with threshold learning starting point at 5 on the WSJ corpus.*

	Value of β (beta)		
Initial threshold	1.4	1.5	1.6
0.012	1,450	1,401	1,404
0.014	1,366	1,360	1,362
0.016	1,419	1,425	1,428
0.018	957	1,401	1,401
0.02	1,097	1,107	1,107
0.022	1,103	1,133	1,121
0.024	1,122	1,365	1,362

Table C.7: *The set F3 of the REPGER algorithm with threshold learning starting point at 10 on the WSJ corpus.*

	Value of β (beta)		
Initial threshold	1.4	1.5	1.6
0.012	1,446	1,390	1,391
0.014	1,414	1,420	1,421
0.016	1,640	1,644	1,645
0.018	1,129	1,431	1,432
0.02	1,021	1,028	1,034
0.022	1,017	1,039	1,027
0.024	1,047	1,302	1,296

Table C.8: *The set F3 of the REPGER algorithm with threshold learning starting point at 15 on the WSJ corpus.*