

# Applying Data Mining to Pseudo-Relevance Feedback for High Performance Text Retrieval

Xiangji Huang<sup>1</sup>, Yan Rui Huang<sup>2</sup>, Miao Wen<sup>2</sup>, Aijun An<sup>2</sup>, Yang Liu<sup>2</sup> and Josiah Poon<sup>3</sup>

<sup>1</sup>School of Information Technology, York University, Toronto, Canada  
jhuang@yorku.ca

<sup>2</sup>Computer Science Department, York University, Toronto, Canada  
{yhuang, mwen, aan, yliu}@cs.yorku.ca

<sup>3</sup>School of Information Technologies, University of Sydney, Sydney, Australia  
josiah@cs.usyd.edu.au

## Abstract

*In this paper, we investigate the use of data mining, in particular the text classification and co-training techniques, to identify more relevant passages based on a small set of labeled passages obtained from the blind feedback of a retrieval system. The data mining results are used to expand query terms and to re-estimate some of the parameters used in a probabilistic weighting function. We evaluate the data mining based feedback method on the TREC HARD data set. The results show that data mining can be successfully applied to improve the text retrieval performance. We report our experimental findings in detail.*

## 1 Introduction

Text classification is the problem of automatically assigning text documents to pre-defined categories. Typically, text classification systems learn models of categories using a large training corpus of labeled data in order to classify new examples. It is well-known, both theoretically [30] and practically [9, 19], that the more labeled training data we have, the more accurate a classification system we get. However, labeled documents are difficult and expensive to obtain, whilst unlabeled documents are plentiful and easy to obtain. How to reduce the demands for labeling data and how to leverage unlabeled data were identified as one of near-term challenges for IR in a recent workshop on Challenges in IR [2].

Co-training is an effective method for utilizing the unlabeled data for classification. In this method, the input data is used to create two predictors that are complementary to each other. Each predictor is then used to classify unlabeled data which is used to train new corresponding complemen-

tary predictors. Typically, the complementarity is achieved either through two redundant views of the same data [5] or through different supervised learning algorithms [12].

Pseudo-relevance feedback (PRF), also known as blind feedback, is a technique commonly used to improve retrieval performance [21, 22, 32]. Its basic idea is to extract expansion terms from the top-ranked documents to formulate a new query for a second round retrieval. Through a query expansion, some relevant documents missed in the initial round can then be retrieved to improve the overall performance. The effect of pseudo-relevance feedback strongly relies on the quality of selected expansion terms from the top-ranked documents. If the words added to the original query are related to the topic, the quality of the retrieval is likely to be improved.

The main focus of this paper is on the use of the data mining technique to boost the performance of pseudo-relevance feedback. In particular, we incorporate a co-training algorithm or a single text classification algorithm without co-training into a retrieval system at the feedback stage. After the initial search, we create a training data set by choosing the top  $k$  passages<sup>1</sup> from the ranked list as positive examples and another  $k$  passages from the bottom of the list as negative examples. The remaining passages are treated as the test set. Then, we classify the examples in the test set and select the most confident examples to add into the labeled training data. If a co-training algorithm is used, this process will be done for a few iterations. The positive examples in the resulting labeled set are used as relevant passages to feed back into the retrieval system.

---

<sup>1</sup>In this paper, the goal of text retrieval is to retrieve relevant passages from a collection of documents given a query, which is called passage level retrieval. The reason why we focus on passage retrieval is that it becomes more and more important in the information retrieval field, which is recognized by TREC conferences [1].

The remainder of the paper is organized as follows. Section 2 outlines the related work. In section 3, we present our search system and show how the data mining technique is integrated into the search system. Section 4 presents our data mining based feedback method in detail. We describe our experiments and results in sections 5, 6 and 7. Finally, we conclude the paper in section 8.

## 2 Related Research

Many methods have been proposed to improve the performance of relevance feedback. Mitra *et al* [22] investigated several ways to improve query expansion by refining the set of documents used in feedback. Fan [11] combined ranking function tuning and blind feedback to improve retrieval performance. Yu [34] proposed a vision-based page segmentation algorithm to assist the selection of query expansion terms in pseudo relevance feedback. Iwayama [18] clustered the initial retrieval result and chose the top N documents from each cluster to feed back into the system. Su [29] presented a new relevance feedback algorithm based on Bayesian classifiers in image retrieval. Yan [33] augmented the retrieval performance by incorporating classification algorithms into PRF and using negative pseudo feedback in video retrieval. He [13] proposed iterative probabilistic one-class SVMs to re-rank the retrieved images in Web image retrieval.

Unlabeled data have been shown to be useful for reducing error rates in text classification [5, 12, 19, 23, 35]. Co-training is used mostly in text classification. It was originally proposed by Blum and Mitchell [5] and used to boost the performance of a learning algorithm when there is only a small set of labeled examples available. Blum and Mitchell [5] show that under the assumption that there are two views of an example that are redundant but not completely correlated, unlabeled data can be used to augment labeled data. Craven uses co-training to extract knowledge from the World Wide Web [10]. The result shows that it can reduce the classification error by a factor of two using only unlabeled data. Kiritchnko and Matwin applied co-training in email classification [20]. In their study, they show the performance of co-training depends on the learning method it uses.

There are a number of other studies that explore the potentials of co-training in recent years. Some modified versions of the co-training algorithm have been proposed since the original one. In [12], Goldman and Zhou use two different supervised learning algorithms to label data for each other. Raskuttii [26] presents a new co-training strategy which uses two feature sets. One classifier is trained using data from original feature space, while the other is trained with new features that are derived by clustering both the labeled and unlabeled data.

All the above works show that the co-training algorithm can be used to boost the performance of a learning algorithm when there is only a relatively small set of labeled examples given. However, this idea is based on the assumption that there are two natural features in the dataset, but a document may not contain two different feature sets. Chan, Koprinska and Poon [8] suggest to randomly split one single feature set into two feature sets for the co-training algorithm. Their experiment results show that a random split of the feature set leads to comparable, and sometimes, even better results than using the natural feature sets.

In this paper, we investigate the effect of integrating Chan *et al*'s version of co-training vs. single classification algorithms without co-training into our search system. The co-training or classification algorithm is used after we obtain the first round result to expand query terms and re-estimate some parameters in the retrieval function.

## 3 The Retrieval System

In this section, we briefly introduce our search system and show how data mining techniques are incorporated into the system architecture. Figure 1 depicts the overall structure of our search system. We use Okapi BSS [3] as the underlying retrieval system, based on which we develop our own index models, query processing functions and feedback modules [17].

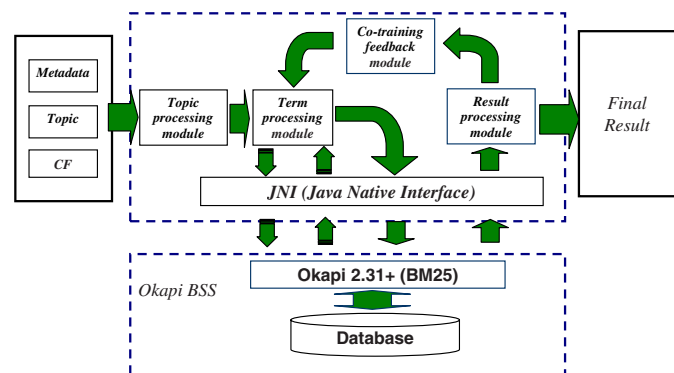


Figure 1. System Architecture

The system takes a query topic in the format provided by TREC [1]. Each topic is associated with some metadata, which include the following fields: Genre, Geography, Granularity, Familiarity, Subject and Related Text<sup>2</sup>. The system also takes two sets of clarification forms (CF) as feedback from users. The topic processing module loads these topic files, extracts relevant fields and converts them

<sup>2</sup>In our experiments, only granularity, geography and related text are used.

into the format acceptable by the term processing module. The term processing module extracts terms from the processed topic in full text documents and conducts the following two tasks. The first task is to extract statistical information of each query term both within the topic and in the full database via Okapi BSS, and then select important terms to query Okapi BSS. The second task is to handle the full text documents provided by the data mining feedback module, expand query terms and update statistics for the existing and new terms.

We build both document-level and passage-level indexes on the database. After receiving a query from the term-processing module, Okapi BSS returns two lists of results. One is passage-based and the other is document-based. The result processing module takes both lists, re-weight the passages based on the two lists and generates a new ranking list of passages<sup>3</sup>. The first  $k$  passages are then passed as blind feedback to the data mining feedback module. After the second round of retrieval, the result processing module returns the final result to the user.

The data mining feedback model takes the blind feedback passages from the result processing module, applies classification algorithms with or without co-training to select more relevant passages that are not in the blind feedback, and passes the new expanded set of feedback passages back to the term processing module. The term processing module then expands query terms, updates statistics for the terms, and issues a new query to Okapi BSS.

Next, we describe the weighting formula used in Okapi BSS. Some of the parameters in the weighting formula will be re-estimated according to the data mining feedback. We then explain how we update the passage-level ranking list according to both document and passage-level retrieval results.

### 3.1 Probability Model

Our underlying search engine, Okapi, is based on the probability model of Robertson and Sparck Jones [27]. The retrieved documents are ranked in the order of their probabilities of relevance to the query. A query term is assigned a weight based on its within-document term frequency and within-query frequency. The weighting function used in Okapi is BM25, shown as follows:

$$\omega = \frac{(k_1 + 1) * tf}{K + tf} * w^{(1)} * \frac{(k_3 + 1) * qtf}{k_3 + qtf} \oplus k_2 * nq * \frac{(avdl - dl)}{(avdl + dl)} \quad (1)$$

$$w^{(1)} = \log \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)} \quad (2)$$

<sup>3</sup>In the experiments of this paper, we focus on passage-level retrieval. Therefore, only the passage-level ranking list is updated by combining the two lists. Documents can also be re-weighted based on the two lists. However, we focus on improving the passage level retrieval performance.

where  $w$  is the weight of a query term,  $N$  is the number of indexed documents in the collection,  $n$  is the number of documents containing the term,  $R$  is the number of documents known to be relevant to a specific topic,  $r$  is the number of relevant documents containing the term,  $tf$  is within-document term frequency,  $qtf$  is within-query term frequency,  $dl$  is the length of the document,  $avdl$  is the average document length,  $nq$  is the number of query terms, the  $k_i$ s are tuning constants (which depend on the database and possibly on the nature of the queries and are empirically determined),  $K$  equals to  $k_1 * ((1 - b) + b * dl / avdl)$ , and  $\oplus$  indicates that its following component is added only once per document, rather than for each term. In our experiments, the values of  $k_1$ ,  $k_2$ ,  $k_3$  and  $b$  in the BM25 function are set to be 1.2, 0, 8 and 0.75 respectively. This parameter setting has been tested on many data sets in different domains, which results in good performance [3, 14, 15, 16, 17, 36].

For each query term,  $R$ , the number of documents known to be relevant for a specific topic, and  $r$ , the number of relevant documents containing the term, are initially set to 0, since the relevant document information is unknown when a query is initially submitted. After the feedback is performed, the value for  $R$  and  $r$  can be updated accordingly. Thus, in the next round of retrieval, BM25 can use the newly updated values of  $R$  and  $r$  to weight terms, which may result in a better list of retrieved documents.

### 3.2 Dual Index Model

In order to obtain more accurate retrieval for passage level inquires, both document level and passage level indexes are built. For each query, we search against both the document level index and the passage level index, and use a merge function to update the weights for passages by combining the results from both indexes. If the passages found in a document are not adjacent, we use the following merge function to assign a new weight to each of these passages:

$$W_{pnew} = (W_p + h_1 * W_d) * \log_{10}(10 * |P|) \quad (3)$$

where  $W_{pnew}$  is the new weight of the passage,  $W_p$  is the weight of the passage we obtain from the passage level index,  $W_d$  is the weight of the document containing the passage, obtained from the document level index,  $|P|$  is the total number of passages retrieved from this document, and  $h_1$  is a coefficient, which is set to be 3 in our experiments

If there are several adjacent passages found in a document, we merge these passages into one and use the following function to assign the weight to the newly merged passage:

$$W_{pnew} = (W_{p_1} + h_1 * W_d) * \log_{10}(10 * |P|) + \frac{1}{2} \sum_{k=1}^n W_{p_k} \quad (4)$$

where  $W_{pnew}$  is the weight of the newly merged passage,  $W_{p_1}$  is the weight of the first of these adjacent passages

from the passage level index,  $W_d$  is the weight of the document we get from the document level index,  $|P|$  is the total number of passages retrieved from this document.  $W_{pk}$  is the weight of the  $k$ th of these  $n$  adjacent passages, and  $h_1$  is a coefficient, which is set to be 3 in our experiments. In [14], we show that the dual index strategy can achieve a very good performance for retrieval at both passage and document levels.

## 4 Feedback with Data Mining

Using the search system described above, we incorporate the data mining technique into the feedback stage of the search. The feedback function provided by Okapi BSS is blind feedback. In blind feedback, the top  $k$  documents<sup>4</sup> returned from the initial search are assumed to be relevant. Based on these  $k$  “relevant” documents, new query terms may be identified and the weight for each query term can be adjusted by updating the values of  $R$  and  $r$  used in Equation (2). Ideally, if these  $k$  documents are truly relevant, a better set of query terms can be formulated and more accurate estimations of  $R$  and  $r$  can be made, which leads to more accurate weighting of query terms. Thus, a better search result can be generated. Therefore, the choice of the  $k$  documents is crucial to the feedback process.

The objective of using data mining in the feedback stage is to improve the quality and quantity of the returned relevant documents so that query terms can be better expanded and  $R$  and  $r$  can be better estimated. The steps of using data mining are as follows. First, we assume the top  $k$  (such as  $k=5$ ) documents in the initial search result are relevant. These  $k$  documents are put into the initial training set for the data mining module and labeled as relevant documents, i.e., positive examples. We then choose another  $k$  documents from the end of the initial search result, put them into the initial training data and label them as non-relevant documents, i.e., negative examples. In this way, we obtain a small set of labeled training data. Then, we apply a data mining algorithm (such as a single classification algorithm or a co-training algorithm) to extend the labeled data by classifying some of the unlabeled data. When this process stops, an extended set of relevant documents is generated and used as feedback documents to expand the query and to re-estimate parameters  $R$  and  $r$ . The number of relevant documents in the extended training data is the new value for  $R$  and the number of such documents containing a query term is the new value for  $r$  for that term.

In order to apply classification and co-training algorithms, we need to represent a document using features.

<sup>4</sup>Since we focus on passage level retrieval, what is provided by blind feedback in our experiments is the top  $k$  passages. However, here and in the rest of the paper, we use the general term **documents** to refer to passages.

Next, we present the feature selection method that we use and our the co-training algorithm in detail.

### 4.1 Document Representation

Extracting features to represent a document is a challenging problem. The most common method for representing a document is to select a “bag of words” as attributes and use the occurrence frequencies of the words as attribute values to represent the document. We use a bag-of-words method to represent documents. However, instead of using occurrence frequencies as attribute values, we exploit a formula proposed in [4] to calculate a weighted frequency for each word and use it as the attribute value for the word. In addition, we propose a word selection method that selects the bag of words based on the query terms and the initial retrieval result from Okapi.

#### 4.1.1 Word Selection

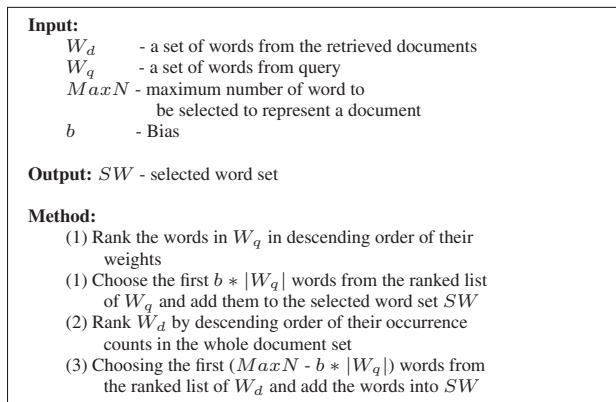
The bag of words are selected from two sets. One, denoted by  $W_d$ , is the set of the words that appear in the documents retrieved by Okapi in the first round. The words in this set are ranked by their occurrence counts in the whole document set. The other set, denoted as  $W_q$ , is the set of words in the query. A word in  $W_q$  is associated with a weight calculated by multiplying its within-query frequency by its BM25 weight returned by Okapi in the initial search. We choose a maximum of  $MaxN$  words from these two sets to represent a document, and use a bias,  $b$  ( $0 \leq b \leq 1$ ), to control the proportion of words chosen from  $W_q$ . Let  $|W_q|$  represent the number of words in  $W_q$ . The algorithm for selecting the bag of words is described in Figure 2. The unique features of this word selection method include that the initial search result is utilized and that query terms and their weights returned by the search engine are considered because they are important in distinguishing between relevant and non-relevant documents. In our experiments,  $b$  is set to 1, meaning that all the words in  $W_q$  are selected. The effect of choosing a value for  $MaxN$  will be discussed in the experiment section.

#### 4.1.2 Word Representation

The selected words are used as attributes to represent a document. The following formula adopted from [4] is used to compute an attribute value to represent a word  $i$  in a document  $j$ :

$$v_{i,j} = (1 - \epsilon_i) \frac{c_{i,j}}{n_j} \quad (5)$$

where  $v_{i,j}$  is the value for word  $i$  and document  $j$ ;  $c_{i,j}$  is the number of times  $i$  occurs in document  $j$ ;  $n_j$  is the total



**Figure 2. The word selection algorithm**

number of words present in document  $j$ ;  $\epsilon_i$  is a normalized entropy of word  $i$  in the whole document set.

Let  $t_i = \sum_j c_{i,j}$ , the total number of times word  $i$  occurs in the whole document set, and  $N$  be the total number of the documents,  $\epsilon_i$  is expressed as :

$$\epsilon_i = -\frac{1}{\log N} \sum_{j=1}^N \frac{c_{i,j}}{t_i} \log \frac{c_{i,j}}{t_i} \quad (6)$$

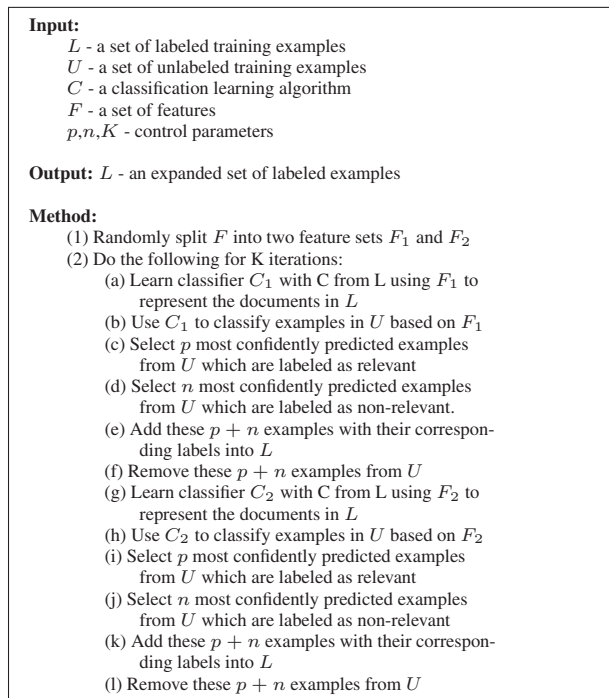
By definition,  $0 \leq \epsilon_i \leq 1$ , where the equalities hold if and only if  $c_{i,j} = t_i$  and  $c_{i,j} = t_i / N$ , respectively. A value of  $\epsilon_i$  close to 1 indicates the word is distributed across many documents throughout the corpus, while a value close to zero means that it only shows in a few documents. Therefore,  $1 - \epsilon_i$  is a measure of selectivity of the word  $i$ .

The effect of  $1 - \epsilon_i$  in Equation 5 is to reflect the fact that two words appearing with the same count in document  $j$  do not necessarily convey the same information about the document. It considers the distribution of the word in the whole document set.

## 4.2 Co-training Algorithm

Having represented documents using features, we are now in the position to describe the co-training algorithm. The algorithm takes a set  $L$  of labeled documents. Half of the documents in  $L$  are the top ranking documents in the initial search result and are labeled as relevant. The other half of the documents in  $L$  are taken from the end of the ranked list and are labeled as non-relevant documents. The remaining documents in the ranked list are considered as a set  $U$  of unlabeled documents. The co-training algorithm iteratively labels some of the unlabeled documents from  $U$ . The algorithm is described in Figure3.

The two classifiers in the co-training algorithm learn from  $L$  based on different sets of features using a learning algorithm. Each classifier supplies new training examples to the other by classifying examples in the unlabeled



**Figure 3. The co-training algorithm**

set. The intuition for this collaborative training to work better than using a single classifier is that if one classifier can find an unlabeled example that is very similar to some of the labeled ones, then it can confidently predict the class of this example and provide one more training example for the other classifier [20]. Because the second classifier uses a different set of features, it may not yield the same labels to the new training examples as done by the first classifier. Therefore, the new example can provide useful information to improve the second classifier. In [5], it is proved that if the two feature sets are conditionally independent given the class and the target classification function is learnable, then any classifier can be boosted to arbitrarily high accuracy using unlabeled examples. Here, we choose the two feature sets by randomly splitting a single feature set. This strategy is chosen based on the findings by Chan *et al* [8]. They found that a random split of a single feature set leads to comparable and, sometimes, even better results than using two natural independent feature sets. In the experiment conducted by Blum and Mitchell [5] for classifying universities' Web pages,  $p$  and  $n$  are set 1 and 3. That is, in each iteration, each classifier is allowed to add 1 new positive and 3 new negative examples to  $L$ . In our experiments, we use similar settings for  $p$  and  $n$  and change the values for other parameters, such as the number of iterations and the learning algorithm used. We will discuss the effect of parameter settings in the experiment section.

### 4.3 Algorithms for Learning

In our studies, we choose three different learning algorithms to test co-training, namely Naive Bayes, decision trees and support vector machines. The results of these three algorithms will be compared.

#### 4.3.1 Naive Bayesian Classifier

The Naive Bayesian classifier is based on Bayes theorem. Suppose that there are  $m$  classes,  $C_1, C_2, \dots, C_m$ . The classifier predicts an unseen example  $X$  as belonging to the class having the highest posterior probability conditioned on  $X$ . In other words,  $X$  is assigned to class  $C_i$  if and only if  $P(C_i|X) > P(C_j|X)$  for  $1 \leq j \leq m$  and  $j \neq i$ . By Bayes theorem, we have

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}.$$

As  $P(X)$  is constant for all classes, only  $P(X|C_i)P(C_i)$  needs to be maximized. Given a set of training data,  $P(C_i)$  can be estimated by counting how often each class occurs in the training data. To reduce the computational expense in estimating  $P(X|C_i)$  for all possible  $X$ s, the classifier makes a naive assumption that the attributes used in describing  $X$  are conditionally independent of each other given the class of  $X$ . Thus, given the attribute values  $(x_1, x_2, \dots, x_n)$  that describe  $X$ , we have

$$P(X|C_i) = \prod_{j=1}^n P(x_j|C_i).$$

The probabilities  $P(x_1|C_i), P(x_2|C_i), \dots, P(x_n|C_i)$  can be estimated from the training data.

#### 4.3.2 Decision Tree

A decision tree is a tree structured prediction model where each internal node denotes a test on an attribute, each outgoing branch represents an outcome of the test, and each leaf node is labeled with a class or class distribution. A typical decision tree learning algorithm adopts a top-down recursive divide-and-conquer strategy to construct a decision tree. Starting from a root node representing the whole training data, the data is split into two or more subsets based on the values of an attribute chosen according to a splitting criterion. For each subset a child node is created and the subset is associated with the child. The process is then separately repeated on the data in each of the child nodes, and so on, until a termination criterion is satisfied.

The most crucial step in decision tree construction for classification is the selection of the attribute on which to branch. An attribute evaluation function can be used for

selecting the best attribute to place at a particular node. The evaluation function typically measures the reduction in class impurity if the attribute were selected for branching. The attribute that produces the lowest impurity partition is selected for branching.

Many decision tree learning algorithms exist. They differ mainly in attribute-selection criteria, termination criteria and post-pruning strategies<sup>5</sup>. Representative decision algorithms include CART [6] and C4.5 [25]. In our study, we use the implementation of the C4.5 decision tree learning algorithm [25] provided in Weka [31].

#### 4.3.3 Support Vector Machine

The support vector machine (SVM) is widely used in text classification in recent years. Its underlying principle is structure risk minimization. Its objective is to determine a classifier or regression function which minimizes the empirical risk (that is, the training set error) and the confidence interval (which corresponds to the generalization or test set error). Given a training data, an SVM determines a hyperplane in the space of possible inputs. This hyperplane will attempt to separate the positives from the negatives by maximizing the distance between the nearest positive examples and negative examples.

In a two-class classification problem, assume that there is a collection of  $n$  training instances  $Tr = \{\mathbf{x}_i, y_i\}$ , where  $\mathbf{x}_i \in \mathcal{R}^N$  and  $y_i \in \{-1, 1\}$  for  $i = 1, \dots, n$ . Suppose that we can find some hyperplane which linearly separates the positive from negative examples in a feature space. The points  $\mathbf{x}$  belonging to the hyperplane must satisfy

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \quad (7)$$

where  $\mathbf{w}$  is normal to the hyperplane and  $b$  is the intercept. To achieve this, given a kernel function  $K$ , a linear SVM searches for Lagrange multiplier  $\alpha_i$  ( $i = 1, \dots, n$ ) in Lagrangian

$$L_p \equiv \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i y_i (x_i \cdot \mathbf{w} + b) + \sum_{i=1}^n \alpha_i \quad (8)$$

such that the margin between two classes, denoted with  $\frac{2}{\|\mathbf{w}\|}$ , is maximized in the feature space [7]. In addition, in the  $\alpha_i$  optimizing process, Karush Kuhn Tucker (KKT) conditions which require

$$\sum_{i=1}^n \alpha_i y_i = 0, \quad (9)$$

<sup>5</sup>Post-pruning is a technique that removes some branches of the tree after the tree is constructed to prevent the tree from over-fitting the training data.

must be satisfied. To predict the class label for a new case  $x$ , we need to compute the sign of

$$f(x) = \sum_{i=1}^n y_i \alpha_i K(x, x_i) + b. \quad (10)$$

If the sign function is greater than zero,  $x$  belongs to the positive class, and the negative otherwise.

In the case of non-separable data, 1-norm soft-margin SVMs minimize the Lagrangian

$$L_p = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i \{y_i(x_i \cdot w + b) - 1 + \xi_i\} - \sum_i \mu_i \xi_i, \quad (11)$$

where  $\xi_i$ ,  $i = 1, \dots, n$  are positive slack variables,  $C$  is a parameter selected by the user with a larger  $C$  indicating a higher penalty to errors, and  $\mu_i$  are Lagrange multipliers introduced to enforce  $\xi_i$  being positive. Similarly, corresponding KKT conditions have to be met for the purpose of optimization. There are several ways to train SVMs. One particularly simple and fast method is Sequential Minimal Optimization (SMO) [24], and we adopt this approach in our study.

## 5 Experimental Setup

We conducted a series of experiments on applying data mining to pseudo-relevance feedback. First, we evaluated the use of single classification algorithms (including naive Bayes, decision tree learning and support vector machines) without co-training. Implementations of those algorithms came from the standard Weka package [31]. In particular, we adopted J48, which accomplished the C4.5 algorithm for decision tree mining [25], in our experiments. We utilized the approach of SMO to train SVMs, and employed Gaussian RBF kernels of the form  $K(x_i, x_j) = \exp(-\gamma|x_i - x_j|^2)$  of C-SVMs. In addition, to obtain the best values of  $\gamma$  and  $C$ , a 10-fold cross validation is conducted on each training dataset. Second, we evaluated the co-training method with different settings of parameters. Third, we compared our data mining based method with two well-known feedback methods reported in [22, 28, 32]. Before presenting the experimental results, we first describe the data set and the performance measure used in the experiments.

### 5.1 Data Sets

The test collection we use is from the HARD track in TREC 2004 [1]. HARD (High Accuracy Retrieval from Documents) is an information retrieval project, the goal of which is to retrieve highly accurate answers to queries by leveraging additional information about the searcher

and/or the search context. This goal is achieved by using techniques like passage retrieval and targeted interaction. The 2004 HARD corpus is around 1.6G. It contains one year (2003) of newswire data from eight sources: AFE (Agence France Presse - English), APE (Associated Press Newswire), CNE (Central News Agency Taiwan), LAT (Los Angeles Times), NYT (New York Times), SLN (Salon.com), UME (Ummah Press - English), and XIE (Xinhua News Agency - English). In total, 635,650 documents and 9,279,957 passages have been indexed in our experiments from the test corpus. 50 evaluation topics were created for the 2004 HARD track. 25 of them are passage based topics and the other are document based topics. In the experiments, we evaluate our method only on the passage retrieval topics since they require higher accuracy and are more sensitive to the relevance feedback. We use 23 of the 25 passage based topics because there are no relevant documents in the database for the other 2 topics according to the official TREC evaluation results.

### 5.2 Performance Measurement

Three character-based measures are used in the 2004 HARD track: Bpref (Binary PREFERENCE relations) at  $X$  characters retrieved, precision at  $X$  characters retrieved, and character-based R-precision. Scores for three different  $X$  values (6000, 12000, and 24000) are reported by LDC<sup>6</sup>. The assumption made by the HARD track in TREC 2004 [1] was that 12000 characters is roughly equivalent to a 100-word passage, so these values are roughly equivalent to evaluating after 5, 10, or 20 (100-word) passages.

The Bpref score for the topic is the average of the scores of its relevant documents. The score for each relevant document is the percentage of non-relevant documents (among the top R non-relevant) that it ranks better than. Bpref tracks closely to MAP (Mean Average Precision) with complete judgments, but degrades much more gracefully than MAP as judgments become more incomplete.

In this paper, we only report the performance accuracy in Bpref, which is also recommended by the 2004 HARD track as the primary passage retrieval measure [1]. All the relevance judgments are provided by LDC. Statistical evaluation on the passage level was done by the script released from the University of Massachusetts and the script from the TREC 2003 evaluation program.

### 5.3 Mitra's Equation

In [22], Mitra *et al.* proposed different ways to refine the set of documents used in feedback. One of the automatic ways is using term correlation information. The equation

<sup>6</sup><http://www ldc.upenn.edu/>

used to compute a new score for each retrieved document is as follows:

$$Sim_{new}D = idf(t_1) + \sum_{i=2}^m idf(t_i) * \min_{j=1}^{i-1} (1 - P(t_i|t_j)) \quad (12)$$

where  $Sim_{new}D$  is the new similarity score,  $idf(t_j)$  is the inverse document frequency of term  $t_i$  if it occurs in document  $D$ , and is 0 otherwise.  $P(t_i|t_j)$  is estimated based on word occurrences in a large set  $S$  of documents initially retrieved for a query and is given by

$$\frac{\text{Number of documents in } S \text{ containing words } t_i \text{ and } t_j}{\text{Number of documents in } S \text{ containing word } t_j} \quad (13)$$

Based on the new similarity scores, the retrieved documents are re-ranked and used in the feedback process to expand the query. Several experiments were conducted in [22]. The results showed that this approach is competitive with the best manual approach. This approach was also compared with Xu's local context analysis [32], and it performed better. We will compare Mitra's results with our data mining based results in Section 6.

#### 5.4 Rocchio's Relevance Feedback

Rocchio proposed a relevance feedback algorithm back in 1971 [28]. It is considered as the standard relevance feedback, and one of the most popular algorithms. The basic idea is that users are asked to evaluate the resulting documents as relevant or non-relevant, and give feedback to the system. Based on the feedback, the system formulates a new query. The equation for computing a weight vector of a new query is illustrated as follows:

$$Q_1 = \alpha * Q_0 + \beta * \sum_{rel} \frac{D_i}{|D_i|} - \gamma * \sum_{nonrel} \frac{D_i}{|D_i|} \quad (14)$$

where  $Q_0$  and  $Q_1$  represent the initial and first iteration query vectors,  $D_i$  represents document weight vectors,  $|D_i|$  is the corresponding Euclidian vector length, and  $\alpha, \beta, \gamma$  are tuning constants. Equation 14 states that for the first round retrieval, the refined query term weights are modified by adding relevant documents term weights and subtract non-relevant documents term weights. In our experiments, we use Equation 14 in a blind feedback process to update query term weights and compare the results with the ones from our data mining based feedback methods.

## 6 Experimental Results

Three classification algorithms Naive Bayes (NB), Decision Trees (DT) and Support Vector Machines (SVM), are used to evaluate the proposed technique.

In terms of the Bpref accuracy, our official submission to the 2004 HARD track, in which only blind feedback is used,

is 0.3576 [14]. This result was the best passage level result among all the official submissions to the 2004 HARD track. The best, medium and worst results for the 2004 HARD track are given in Table 1. This result will be used as the base result in our experiments, which will be compared to all the results generated by applying data mining to pseudo-relevance feedback.

	Best	Medium	Worst
Bpref@12K	0.3576	0.222	0.048

**Table 1. The 2004 HARD Track Performance**

### 6.1 Results for Single Classifiers

We conducted a series of experiments under different settings using a single classifier (NB, DT or SVM) without co-training. For each run, we choose the top 5 documents from the initial retrieval result as positive examples and bottom 5 documents as the negative examples. These 10 documents constitute the training set and the remaining documents constitute the test set.<sup>7</sup> For each classification algorithm, a classifier is built based on the training data and is used to classify the documents in the test set. We then select  $k$  documents in the test set which are considered by the classifier to be most probably relevant. These  $k$  documents plus the top 5 documents returned from the first round retrieval, which makes 11 documents in total matching the number of feedback documents in the base run, are fed back into the retrieval system for query expansion and parameter estimation. Table 2 shows the result from the second round of retrieval. For each classification algorithm, we have a few runs with different  $k$  and  $MaxN$  values. Table 2 shows the average results for each algorithm in terms of Bpref value at 12,000 characters on the basis of 14 experiments with the same experimental settings for co-training. The value in the parentheses is the relative rate of improvement over the base run. The last row shows the standard deviation of the runs for each algorithm.

	NB	DT	SVM
Average	0.3793 (6.07%)	0.3854 (7.77%)	0.3825 (6.96%)
STD	0.023	0.017	0.013

**Table 2. Single classifiers without co-training**

From Table 2, we can see that using a single classifier to re-rank the documents improves the retrieval performance compared to the base run whose result is 0.3576. Among

<sup>7</sup>We use only the top 5 documents as positive examples in our training set because for some topics in the 2004 HARD track only a few documents are relevant. Our experiments showed that enlarging the training set by including more top-ranking documents would reduce the performance.

three algorithms, DT achieves the best average result. SVM is a close second and is most stable.

To show how the values for  $k$  and  $MaxN$  affect the retrieval result, Figure 4 depicts the performance of the three classifiers with different  $MaxN$  values given  $k=6$ , and Figure 5 depicts the performance with different  $k$  values given  $MaxN=300$ . From the figures, we can observe that DT and SVM are less sensitive to  $MaxN$  than NB. For NB, a larger number of features generally leads to better performance than a smaller number of features. In terms of  $k$ , the value of 2 leads to the best performance for all the classifiers. When  $k$  increases, the performance decreases. Especially, the performance decrease is significant for NB and DT.

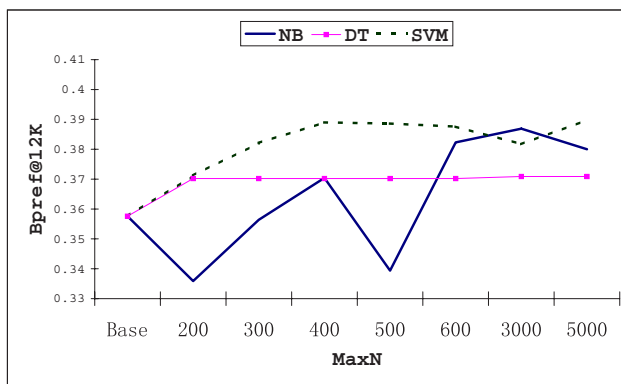


Figure 4. Single Classifiers with  $k=6$

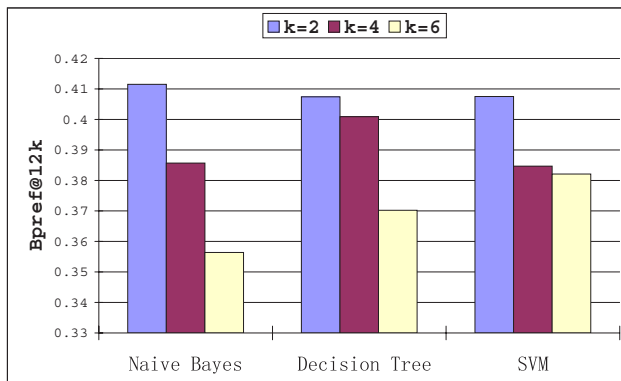


Figure 5. Single classifiers with  $MaxN=300$

## 6.2 Results for Co-training

We now use co-training with each of the three classification algorithms (NB, DT and SVM). To make it practically

feasible, the number of positive and negative examples used for co-training data is usually small. In this experiment, to formulate the first round co-training set, we select five top 5 documents (as positive examples) and bottom five documents (as negative examples) from the initial retrieval result. For each classification algorithm, a series of 14 experiments are conducted using co-training with different values for iteration number  $K$  and  $MaxN$ . Table 3 shows the performance for each classifier in terms of the Bpref measure and relative rate of improvement over the base run. Comparing Table 3 with Table 2, we can find that co-training improves the results for all the three classifiers NB, DT and SVM. Among all the results in Tables 2 and 3, using co-training with the decision tree learning method achieves the best result. We can also see that co-training leads to more stable results for all the three classification algorithms compared with using the single classifier method without co-training.

	NB	DT	SVM
Average	0.3867 (8.14%)	0.3998 (11.8%)	0.3969 (10.99%)
STD	0.013	0.0035	0.0041

Table 3. Co-training Performance

To illustrate the results graphically, we re-plot these data in Figure 6, in which the x-axis represents 14 experiments plus the base run and the y-axis shows the retrieval performance in terms of Bpref at 12,000 characters. An experiment is represented by  $WiKj$ , which means  $K$  is set to  $j$  and  $MaxN$  is set to  $i \times 100$ . For example,  $W3K2$  stands for the experiment in which  $MaxN$  is set to 300 and  $K$  is set to 2. From Figure 6, we can see that all the results are better than the base result, and DT performs the best among the three classifiers in most of the cases. The highest accuracy in terms of Bpref@12K is achieved by NB when  $MaxN = 1000$  and  $K = 2$ , even though NB's performance is not as stable as the performance of DT and SVM.

To show how the values of  $MaxN$  influences the retrieval result, Figure 7 depicts the retrieval performance resulting from using co-training feedback at  $K = 3$  with different  $MaxN$  values. We can observe that the decision tree and support vector machine classifiers are not sensitive to the change of the  $MaxN$  value. But Naive Bayes classifier is sensitive to the  $MaxN$  value.

To show how the number of co-training iterations ( $K$ ) influences the result, Figure 8 depicts the retrieval performance resulting from using co-training feedback at  $MaxN = 300$  with different  $K$  values. With a fixed  $MaxN$  value, the retrieval performance resulting from using co-training feedback depends on its  $K$  value. The best retrieval result can be achieved by setting  $K$  to 3 for Naive Bayes and deci-

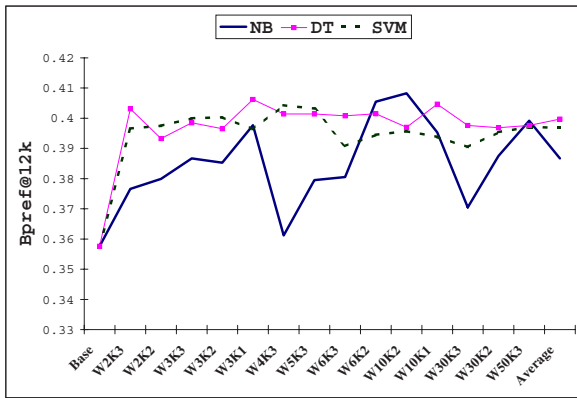


Figure 6. Performance of Co-training

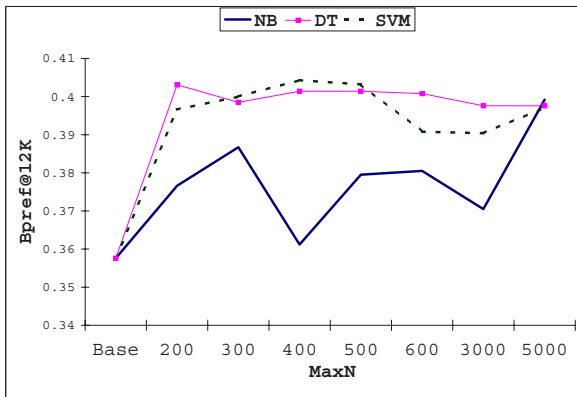


Figure 7. Performance with  $K = 3$

sion tree classifiers<sup>8</sup>. The retrieval performance from using support vector machine is pretty stable and not sensitive to the change of the  $K$  value.

### 6.3 Comparison with Mitra’s Method

Using Mitra’s equation in Section 5.3, we re-rank the initial retrieved documents. The newly ranked documents are used in the blind feedback process for query expansion. Table 4 shows the results for the Mitra’s method. In the experiments, we use both “Title” and “Description” fields. Top N indicates the number of re-ranked documents for query expansion. From the experimental results, we can see that

<sup>8</sup>In each iteration of co-training, each of the two trained classifiers adds one relevant document and two non-relevant documents to the training set. By setting  $K$  to 3, there are 6 relevant documents returned by the co-training algorithm plus the top 5 relevant documents from the blind feedback. Thus, there are totally 11 documents marked as relevant documents from the data.

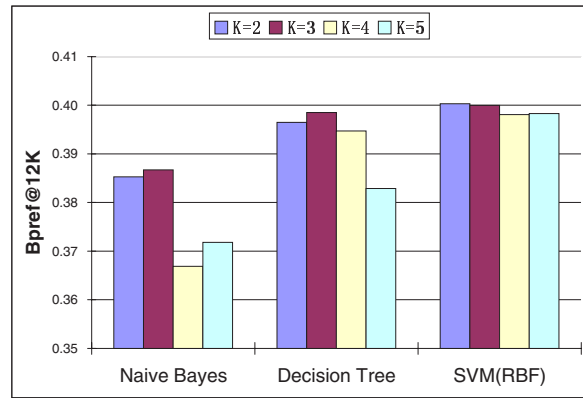


Figure 8. Performance with MaxN=300

	Top 100	Top 50	Top 20
Bpref@12K	0.3554	0.3606	0.3596

Table 4. Performance of Mitra’s Method

Mitra’s equation does not perform as well as our feedback methods that use single classifiers or co-training.

### 6.4 Comparison with Rocchio’s Method

Using Rocchio’s equation in Section 5.4, we use the blind feedback to formulate a new query. We assume that the top  $K$  documents are relevant and the bottom  $K$  documents are not relevant, where  $K$  is set to be 11 in our experiments. The results are illustrated in Table 5. In this table, passage level evaluation in terms of Bpref@12K is used. Rocchio’s Feedback 1 is the blind feedback which only considers the relevant documents and use these documents to expand the query list. Rocchio’s Feedback 2 is the blind feedback which uses both relevant and non-relevant documents to expand the query list. We set  $\alpha$ ,  $\beta$ , and  $\gamma$  to 1 respectively. From Table 5, we can see that the performance of Rocchio’s feedback method is better than the base run, but not as good as our methods. We also notice that its performance degrades by considering non-relevant documents. Our conjecture is that terms contained in relevant documents are more important than those provided in non-relevant documents. The non-relevant documents may provide some noise information in the term re-weighting process.

## 7 Analysis and Discussion

The above results are the average results over all the 23 topics. To see whether the improvement of the retrieval

	Rocchio's Feedback 1	Rocchio's Feedback 2
Bpref@12K	0.3783	0.3691

**Table 5. Performance of Rocchio's Method**

performance is due to the better labeling of documents by data mining (or whether the decrease in retrieval performance is due to the worse labeling of documents by data mining), we analyze the co-training results for three topics. For two of the three topics co-training makes a big retrieval improvement, and for the third topic co-training degrades the retrieval performance. For a run with co-training, we collect the classification accuracy of the co-training algorithm, which is the percentage of relevant documents that are identified by co-training. If we compare this classification accuracy with the percentage of relevant documents in the base run's feedback list without the top 5 documents, we can know whether the co-training feedback provides more relevant documents to the search engine than the blind feedback. Table 6 compares the co-training runs with the base run in terms of both Bpref@12K and the classification accuracy (denoted as "Acc."). For the base run, "Acc" means the percentage of relevant documents in the base run's feedback list without the top 5 documents. For a run with co-training such as DT, "Acc" means the percentage of relevant documents that are identified by co-training.

We can see that the retrieval performance for these three topics changes dramatically by using co-training. For the topics 407 and 445, their retrieval performance increases compared to the base one. But for topic 415, their retrieval performance decreases. For topic 407, both DT and SVM provide more relevant documents to the feedback list, which leads to the increase in retrieval performance. For the same topic, NB does not provide more relevant documents to the feedback list, but its retrieval performance is still improved significantly. Similar situations happen to topic 445, in which SVM identified more relevant documents for the feedback list and results in a better retrieval result, while DT and NB do not provide more relevant documents to the feedback list but still improve retrieval performance. One explanation to this phenomenon is that even though co-training sometimes cannot supply more relevant documents to the feedback list, it may provide documents that contain better search terms so that when the query is expanded with these terms, the retrieval result can be improved. For topic 415, its performance decreases compared to the base one. This is because co-training algorithms return more non-relevant documents than the base one.

Regarding the efficiency of co-training, it is obvious that the use of co-training takes more time than pseudo-relevance feedback. However, since the size of the training set and the number of iterations used in co-training are both

Topics	Base		DT		SVM		NB	
	Bpref	Acc.	Bpref	Acc.	Bpref	Acc.	Bpref	Acc.
407	0.13	0%	0.62	25%	0.42	27%	0.61	0%
415	0.72	100%	0.41	35%	0.15	52%	0.60	80%
445	0.20	0%	0.58	0%	0.56	7.64%	0.58	0%

**Table 6.** Retrieval performance vs. classification accuracy

small, the increase in running time is not much.

## 8 Conclusions

We have presented a new feedback method that combines data mining and pseudo-relevance feedback to enhance the retrieval performance. The proposed method is evaluated on the TREC 2004 HARD dataset. Our results demonstrate that data mining can be successfully applied to information retrieval, especially when there is no labeled example or labeled examples are very few.

The contributions of this paper are as follows. First, we show in detail how to apply the data mining technique to the feedback process of information retrieval. Second, we demonstrate that the data mining based feedback method is effective in improving retrieval performance. Third, we show that for both single classifier based and co-training based methods, DT and SVM perform the best and are the least sensitive to parameter settings. Fourth, we show that co-training can achieve better performance than using single classifiers. In addition, for all the three classification algorithms (DT, SVM and NB), co-training leads to more stable results. Fifth, we compare our data mining based method with the popular automatic feedback method reported by Mitra *et al.* and the standard feedback method reported by Rocchio. We find that our method performs the best. Furthermore, by looking into the results for individual topics, we find that if co-training provides a feedback list with more relevant documents, the retrieval performance can be improved greatly. However, sometimes, even if it does not identify more relevant documents, retrieval performance can still be improved. Our conjecture is that the documents identified by co-training through text classification can supply better terms for query expansion even though they are not "relevant". We plan to investigate this issue further. Our future work also include investigating how to automatically choose parameter  $k$  for single classifiers or parameter  $K$  for co-training. The parameter determines the number of documents used in feedback. We conjecture that the value of  $k$  or  $K$  should depend on the topic. We also consider designing a hybrid classifier for the co-training algorithm, which may further improve the retrieval performance.

## Acknowledgments

This study was supported by a research grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada. We thank three anonymous reviewers for their useful comments on the paper.

## References

- [1] J. Allan. HARD Track Overview in TREC 2004. In *Proceedings of TREC-13*. NIST Special Publication, 2004.
- [2] J. Allan and et al. Challenges in IR and Language Modeling. *SIGIR Forum*, 37(1):31–47, 2003.
- [3] M. Beaulieu, M. Gatford, X. Huang, S. Robertson, S. Walker, and P. Williams. Okapi at TREC-5. In *Proceedings of TREC-5*, pages 143–166, 1997.
- [4] J. Bellegarda. Exploiting Latent Semantic Information in Statistical Language Modeling. *Proceedings of the IEEE*, 88(8):1279 – 1296, 2000.
- [5] A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proc. of the Workshop on Computational Learning Theory*, pages 92–100, 1998.
- [6] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
- [7] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [8] J. Chan, I. Koprinska, and J. Poon. Co-training with a Single Natural Feature Set Applied to Email Classification. In *Proc. of Australasian Doc. Comput. Symposium*, pages 47–54, 2004.
- [9] W. Cohen. Learning Rules that Classify Email. In *Proc. of the AAAI Spring Symposium on Machine Learning in Information Access*, pages 18–25, 1996.
- [10] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, 118(1/2):69–113, 2000.
- [11] W. Fan, M. Luo, L. Wang, W. Xi, and E. A. Fox. Tuning Before Feedback: Combining Ranking Discovery and Blind Feedback for Robust Retrieval. In *Proceedings of the 27th Annual International ACM-SIGIR*, pages 138–145, 2004.
- [12] S. Goldman and Y. Zhou. Enhancing Supervised Learning with Unlabeled Data. In *Proc. 17th ICML*, 2000.
- [13] J. He, M. Li, Z. Li, H. Zhang, H. Tong, and C. Zhang. Pseudo Relevance Feedback Based on Iterative Probabilistic One-Class SVMs in Web Image Retrieval. In *Proceeding of Pacific-Rim Conference on Multimedia*, 2004.
- [14] X. Huang, Y. Huang, and M. Wen. A Dual Index Model for Contextual IR. In *Proc. of the 28th ACM SIGIR*, 2005.
- [15] X. Huang, Y. Huang, M. Wen, and M. Zhong. York University at TREC 2004: Genomics and HARD Tracks. In *Proceedings of TREC-13*. NIST Special Publication, 2004.
- [16] X. Huang, F. Peng, D. Schuurmans, N. Cercone, and S. Robertson. Applying machine learning to text segmentation for information retrieval. *Information Retrieval Journal*, 6(4):333–362, 2003.
- [17] X. Huang, M. Wen, A. An, and Y. Huang. A Platform for Okapi-Based Contextual Information Retrieval. In *Proc. of the 29th ACM SIGIR*, 2006.
- [18] M. Iwayama. Relevance Feedback with a Small Number of Relevance Judgements: Incremental Relevance Feedback vs. Document Clustering. In *Proceedings of the 23rd annual international ACM SIGIR*, pages 10–16, 2000.
- [19] T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proc. of the 16th ICML*, pages 200–209, 1999.
- [20] S. Kiritchenko and S. Matwin. Email Classification with Co-training. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*, 2001.
- [21] T. Lynam, C. Buckley, C. Clarke, and G. Cormack. A Multi-System Analysis of Document and Term Selection for Blind Feedback. In *Proc. of CIKM'04*, pages 261–269, 2004.
- [22] M. Mitra, A. Singhal, and C. Buckley. Improving Automatic Query Expansion. In *Proceedings of the 21st International ACM-SIGIR*, pages 206–214, 1998.
- [23] K. Nigam, A. McCallum, S. Thrun, and M. T. Text Classification from Labeled and Unlabelled Documents Using EM. *Machine Learning*, 39(2/3):103–134, 2000.
- [24] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, pages 185–208, 1999.
- [25] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [26] B. Raskutti, H. Ferra, and A. Kowalczyk. Combining Clustering and Co-training to Enhance Text Classification Using Unlabelled Data. In *Proceedings of the 8th Int. Conf. on KDD*, pages 620–625, 2002.
- [27] S. Robertson and J. Sparck. Relevance Weighting of Search Terms. *JASIS*, 27(3):129–146, 1976.
- [28] J. J. Rocchio. Relevance Feedback in Information Retrieval. In *In the SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall Inc., 1971.
- [29] Z. Su, H. Zhang, and S. Ma. Using Bayesian Classifier in Relevant Feedback of Image Retrieval. In *Proc. of IEEE International Conf. on Tools with Artificial Intelligence*, 2000.
- [30] L. Valiant. A Theory of the Learnable. *Communication of the ACM*, 27(11):1134–1142, 1984.
- [31] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.
- [32] J. Xu and W. B. Croft. Query Expansion Using Local and Global Document Analysis. In *Proceedings of the 19th International ACM SIGIR*, pages 4–11, 1996.
- [33] R. Yan, A. Hauptmann, and R. Jin. Negative Pseudo-Relevance Feedback in Content-based Video Retrieval. In *Proceedings of ACM Multimedia 2003*, 2003.
- [34] S. Yu, D. Cai, J. Wen, and W. Ma. Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation. In *Proc. of the 12th International Conference on World Wide Web*, pages 11–18, 2003.
- [35] S. Zelikovitz and H. Hirst. Using LSI for Text Classification in the Presence of Background Text. In *Proc. of the 10th CIKM*, pages 113–118, 2001.
- [36] M. Zhong and X. Huang. Concept-Based Biomedical Text Retrieval. In *Proc. of the 29th ACM SIGIR*, 2006.