

# A Text Filtering Method for Digital Libraries

Hayri Sever and M.Zafer Bolat  
Department of Computer Engineering  
Baskent University  
06530, Bağlıca, Ankara, Turkey  
{ sever, zafer@baskent.edu.tr }

***Abstract.** Since information filters have become essential mediators between information sources and their users, information filtering is an important component of modern information systems. The objective of information filtering is to judge the documents as “relevant or non-relevant to users” according to user needs as documents arrive into the system. In this paper, we investigate an information filtering method based on steepest descent induction algorithm, which is an enhanced version of a relevance feedback algorithm, combined with a two-level preference relation on user ranking. The performance of the proposed algorithm is experimentally evaluated. The experiments are conducted using publicly available Reuters-21578 Distribution 1.0 data collection which is a well known dataset in text categorization and machine learning experiments as corpus. A microaverage breakeven effectiveness measure is used for performance evaluation. The best size of negative data employed in the training set is empirically determined and the effect of  $R_{norm}$  factor on the learning process is evaluated. Finally, we demonstrate effectiveness of proposed method by comparing experimental results to five inductive learning methods. Our method outperforms Findsim, it competes with Naive Bayes, Bayes Nets and Decision Trees. We found that Linear Support Vector Machines is the best among others.*

## 1 Introduction

More and more users are faced with the problem of selecting relevant information in a space of different information-sources. To overcome this *information overload problem* [1] information filtering (IF) techniques are being developed to deliver information to users. Information filtering is the process that monitors documents as they enter the system and selects only those that match the user query (known as a *profile*). Information retrieval (IR), on the other hand, accepts user needs (requests) as a query and provides the user with a list of documents ordered (ranked) based on the similarity with the user query.

In IR, the term linear separation is used for retrieval strategies that partitions relevant and non-relevant documents into distinct blocks. Linear separation can be simply implemented by constructing optimal queries that rank (list) relevant documents higher than non-relevant. By importing the same notion to IF, we define an optimal query to be a query that most likely classifies a new document as relevant or non-relevant with respect to some classification function minimizing the error. In this paper, we propose a framework for the filtering problem based on *Steepest Descent Algorithm* SDA [8], combined with a two-level preference relation on user ranking. The system is trained on pre-judged documents and an optimal query is formulated following the proposed framework.

## 2 Preliminaries

A typical information retrieval systems  $S$  can be defined as a 5-tuple,  $S=(T,D,Q,V,f)$  where:  $T$  is a set of ordered index terms,  $D$  is a set of documents,  $Q$  is a set of queries,  $V$  is a subset of real numbers, and  $f:D \times Q \Rightarrow V$  is a retrieval function between a query and a document [4]. IR systems based on the vector processing model represent documents by vectors of term values of the form  $d=(t_1, w_{d1}; t_2, w_{d2}; \dots; t_n, w_{dn})$ , where  $t_i$  is an index term in  $d$  (i.e.  $t_i \in T \cap d$ ) and  $w_{di}$  is the weight of  $t_i$  that reflects relative importance of  $t_i$  in  $d$ . Similarly, for a query  $q \in Q$ , it is represented as  $q = (q_1, w_{q1}; q_2, w_{q2}, \dots; q_m, w_{qm})$ , where  $q_i \in T$  is an index term in  $q$  (i.e.  $q_i \in T \cap q$ ) and  $w_{qi}$  is the weight of query term  $q_i$  that reflects relative importance of  $q_i$  in  $q$ . Our objective is to formulate an optimal query,  $Q_{opt}$ , that discriminates more preferred documents from less preferred documents. With this objective in mind, we define a preference relation  $\succeq$  on a set of documents, in a retrieval (ranking) output as follows. For  $d, d' \in \Delta$ ,  $d \succeq d'$  is interpreted as  $d$  is preferred to or is equally good as  $d'$ .

The essential motivation is that  $Q_{opt}$  provides an acceptable retrieval output; that is, for all  $d, d' \in \Delta$ , there exists  $Q_{opt} \in Q$  such that  $d \succeq d' \Rightarrow f(Q_{opt}, d) > f(Q_{opt}, d')$ . A performance measure may be derived by using the distance between a user ranking and a system ranking. A possible evaluation measure is  $R_{norm}$  as suggested by Bollmann and Wong [3], other measures have also been proposed [9]. Let  $(D, \succeq)$  be a document space, where  $D$  is a finite set of documents and  $\succeq$  be a preference relation as defined above. Let  $\Delta$  be some ranking of  $D$  given by a retrieval system. Then  $R_{norm}$  is defined as

$$R_{norm}(\Delta) = \frac{1}{2} \left( 1 + \frac{S^+ - S^-}{S_{max}^+} \right)$$

where  $S^+$  is the number of document pairs where a preferred document is ranked higher than non-preferred document,  $S^-$  is the number of document pairs where a non-preferred document is ranked higher than preferred one, and  $S_{max}^+$  is the maximal number of  $S^+$ .

In this paper  $Q_{opt}$  is formulated inductively by SDA as described in [8].

Let  $B = \{b = d - d' : d \succeq d'\}$  be the set of difference vectors in an output ranking. To obtain  $Q_{opt}$  from any query  $Q$ , we solve  $f(Q, b) > 0$  for all  $b \in B$ . It is assumed here that  $f(Q, d) = Q^T d$ , which is the cross product of vectors, and for  $f(Q, d) > f(Q, d') \Rightarrow Q^T d > Q^T d' \Rightarrow Q^T (d - d') > 0 \Rightarrow f(Q, b) > 0$ . The steps of the algorithm are as follows.

1. Set starting query vector  $Q_0$ ; let  $k = 0$ .
2. Let  $Q_k$  be a query vector at the start of the  $(k+1)th$  iteration; identify the following set of difference vectors:

$$\Gamma(Q_k) = \{b = d - d' : d \succeq d' \text{ and } f(Q_k, b) \leq 0\};$$

if  $\Gamma(Q_k) = \emptyset$ ,  $Q_{opt} = Q_k$  is a solution and exit, otherwise,

3. Let

$$Q_{k+1} = Q_k + \sum_{b \in \Gamma(Q_k)} b$$

4.  $k = k+1$ ; go back to Step (2).

Predicted Label	Actual Label	
	Relevant	Non – Relevant
Relevant	a	b
Non-relevant	c	d

(a)

	training	Test
With at least one topic	7775	3019
With no topic	1828	280
Total	9603	3299

(b)

**Fig. 1.** (a) Measures of system effectiveness. (b) Number of documents in the collection.

### 3 Effectiveness Measures

Consider a system that is required to categorize  $n$  documents by a query; the result is an outcome of  $n$  binary decisions. From these decisions a contingency table is constructed as in Figure 1(a). Each entry in the table specifies the number of documents with the specified outcome label. In our experiment, the performance is measured with *precision* and *recall*. Precision is defined as the percentage of relevant documents correctly retrieved among the whole retrieved set, and recall is the percentage of relevant documents retrieved among the whole set of relevant documents stored in the system. For text categorization, precision =  $a/(a + b)$  and recall =  $a/(a + c)$ . We use *Microaverage* [6] for effectiveness measure. Consider a system with  $n$  documents and  $q$  queries. Then there are  $q$  contingency tables similar to the one in Figure 1(a) representing the decisions of the queries when evaluated against all  $n$  documents. *Microaverage*, adds up the  $q$  contingency tables all together, and then precision and recall are computed. The point where precision and recall are absolutely equal is called as *breakeven* point

### 4 The Experiment

In this section, we describe the experimental method in detail.

#### 4.1 Reuters-21578 Data Set and Text Representation

To experimentally evaluate the proposed information filtering method we use Reuters-21578 text categorization test collection distribution 1.0 as our corpus [10]. This collection consists of 21,578 documents, that are Reuters newswire stories. The documents of this collection are divided into training and test sets We restrict our study to only *TOPICS* category, used the Modified Apte split, screened out documents without topics. Figure 1(b) shows some statistics about the number of

documents in the collection. We pre-process the train and unused documents, performing parsing and tokenizing the text portions, producing a dictionary of single words excluding numbers [2]. We used a universal list of 343 stop words to eliminate words with little discriminatory power from the dictionary [11]. The Porter stemmer algorithm was employed to reduce each remaining words to word-stems form [12]. Since any word occurring only a few times is statistically unreliable [7], the words occurring less than five times are eliminated. The words are then sorted in descending order of frequency.

Our experiments are based on the Vector Space Model (VSM). In this model documents and queries are represented as vectors of weighted terms. One common function for computing the term weights is  $w_k = t_k \times \log(N_D/n_k)$  [7], where  $t_k$  is the term frequency,  $N_D$  is the total number of documents in the collection, and  $n_k$  is the number of documents containing  $t_k$ . Since long documents have higher term frequencies and uses more terms, we used cosine normalization.

$w'_k = w_k / \sqrt{\sum w^2}$  where  $w'_k$  is unnormalized weight of term  $k$ , and  $w_k$  is normalized weight of term  $k$

#### **4.2 Training**

We have neither a retrieval output nor a user query. Instead, we have a number of topics and for each topic the document collection is partitioned into training and test cases. The training set contains only positive examples of a topic. In this sense, the training set is not a counterpart of the retrieval output due to the fact that we do not have negative examples. We can, however, construct a training set for a topic that consists of positive and negative examples, under the plausible assumption that any document considered as positive example for the other topics and not in the set of positive examples of the topic at hand is a candidate for being a negative example of this topic. The size and especially the quality of the training set is an important issue in generating an induction rule set.

We include the entire set of positive examples because the SDA with a larger number of positive examples makes the algorithm produce more efficient induction rules. Additionally, the result published by Dumais et al. [5] for the Reuters-21578 data shows that the performance degrades when samples of the positive examples are considered in the training set. The negative data, on the other hand, is sampled because considering the entire set of negative data of a topic increases the learning process overhead and may force the solution vector away from the solution region. For the purpose of estimating the best size for negative data in the training set, we conducted preliminary experiments. The best performance on the top 16 topics is obtained when the proportion of negative data is in the range 50%-80% of the positive data. Therefore, we fix the size of negative sample to 50% of that of the positive set.

We set starting query as mean of positive examples. Within the algorithm loop we continually update the query,  $Q_k$ , that yields the highest  $R_{norm}$  value (1.0) in order to

return  $Q_k$  as optimal query. If  $R_{norm}$  value does not reach 1.0 in maximum 150 iterations the algorithm is terminated and the query with the highest  $R_{norm}$  value is set as optimal query.

### 4.3 Results

The choice of  $R_{norm}$  value is important in the learning process. A higher value may produce a better query on the expense of increasing the processing overhead. Finally, after all the necessary parameters are fixed, we train the system on all of the 118 topics.

**Table 2. Comparing results with other five inductive algorithms. Breakeven in % is computed on top 10 topics and on overall 118 topics.**

Topic	Findsim	NBayes	SDA	BayesNets	Trees	SVM
earn	92,9	95,9	96,32	95,8	97,8	98,0
acq	64,7	87,8	85,26	88,3	89,7	93,6
money-fx	46,7	56,6	68,72	58,8	66,2	74,5
grain	67,5	78,8	71,81	81,4	85,0	94,6
crude	70,1	79,5	82,54	79,6	85,0	88,9
trade	65,1	63,9	65,25	69,0	72,5	75,9
interest	63,4	64,9	61,07	71,3	67,1	77,7
wheat	68,9	69,7	76,06	82,7	92,5	91,9
ship	49,2	85,4	65,17	84,4	74,2	85,6
corn	48,2	65,3	75,00	76,4	91,8	90,3
<b>Avg.Top.10</b>	<b>64,6</b>	<b>81,5</b>	<b>84,54</b>	<b>85,0</b>	<b>88,4</b>	<b>92,0</b>
<b>Avg.All Cat.</b>	<b>61,7</b>	<b>75,2</b>	<b>76,37</b>	<b>80,0</b>	<b>NA</b>	<b>87,0</b>

*NBayes, BayesNets, Trees, and SVM methods are Naive Bayes, Bayes Nets, Decision Trees, and Linear Support Vector Machines methods, respectively. For further details of these methods the reader is referred to [5].*

As a comparative study, Table 2 presents results of SDA and other five inductive algorithms that were recently experimented on Reuters-21578 dataset [4]. SDA outperforms Findsim, and is almost as good as NBayes, BayesNet, and Trees methods. It is however outperformed by Linear SVM method due to the fact that relevance feedback methods (including SDA) require as large size of positive data as possible for drifting the query towards the solution region. Therefore, for topics with small number of positive examples, which represent the majority of the topics in the Reuters-21578 data, the optimal query which is close to the solution region is hard to find and the performance of the algorithm SDA is the same as Findsim method on these topics. Nevertheless, on average it outperforms the Findsim method by a significant margin which upholds the plausible fact that higher-order approximation methods, such as SDA, outperforms their counterpart first-order approximation methods, such as Findsim.

## 5 Conclusion

When the SDA is compared to other inductive algorithms experimented on Reuters-21578, it outperforms Findsim method. However, it competes with NBayes, and BayesNet, Trees methods and outperformed by Linear SVM method. Since information filtering is a dynamic process which keeps updating the query/classifier incrementally based on incoming documents, in future work we intend to investigate application of incremental learning to information filtering.

### References

1. G. Fischer, C. Stevens, *Information Access in complex, poorly structured information spaces, Proceedings CHI conference*, pages.63-70, April 1991.
2. Apt C., Damerau F., and Weiss S. M. *Automated learning of decision rules for text categorization. ACM Transactions on Information Systems*, 12(3):233-251, July 1994.
3. Bollman, P. and Wong, S. K. M. *Adaptive linear information retrieval models. In Proc. of the 10th Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 157-163, New Orleans, LA., June 1987.
4. Bookstein, A., and Cooper, W. *A general mathematical models for information retrieval systems. Library Quarterly*, 46(2):153-167, 1976.
5. Dumais, S., Platt, J., Heckerman, D., and Sahami, M. *Inductive learning algorithms and representations for text categorization. In Proceedings of ACM-CIKM98*, Nov.1998.
6. Lewis, David D. *Evaluating text categorization. In Proceedings of Speech and Natural Language Workshop*, pages 312-318, February, 1991.
7. Salton, G, *Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesely*, 1988.
8. Wong, S. K. M., and Yao, Y. Y. *Query formulation in linear retrieval models. Journal of the American Society for Information Science*, 41, 5 (1990), 334-341.
9. Wong, S. K. M. *Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information Science*, 46(2):133-145, 1995.
10. Reuters-21578 collection is available at: <http://www.research.att.com/~lewis>
11. The stop list is available at: [http://www.iiasa.ac.at/docs/R\\_Library/libsrchs.html](http://www.iiasa.ac.at/docs/R_Library/libsrchs.html).
12. The source code for the Porter Algorithm is found at: <http://ils.unc.edu/keyes/java/porter/index.html>.