Improving Learning Accuracy in Information Filtering

H.C.M. de Kroon[1,2] , T.M. Mitchell[1] , E.J.H. Kerckhoffs[2]

[1] Carnegie Mellon University
School of Computer Science
Pittsburgh, USA
email: {edekroon|tom.mitchell}@cs.cmu.edu

[2] Delft University of Technology
Faculty of Technical Mathematics and Informatics
Delft, the Netherlands
email: eugene@kgs.twi.tudelft.nl

Abstract

The increasing stream of electronic information available makes it ever more time consuming to find interesting information. Information filtering systems that use machine learning techniques hold the promise of relieving users of this burden by learning a model of the user's interests and using this model to find interesting documents. Most of these systems however suffer from the problem that users must use the system for a long time before a useful model of their interests can be learned. This paper considers how this delay might be reduced by taking advantage of information such as the user's World Wide Web homepage and bookmarks. Autoclass, an algorithm for automatic classification, has been used to find clusters within this extra information. These clusters have been used to get a 5% to 30% improvement in learning accuracy.

1 Introduction

An ever increasing stream of electronic information comes to us every day. Not everything in this stream is of interest, but the really interesting things can only be found by going through everything. Information filtering techniques try to relieve users of this burden by keeping a model of the user's interests. This model, called a *profile* in information filtering, is used to automatically filter out the information that the user probably doesn't want to see.

An information filter has to maintain one profile for a user for a longer period of time, and therefore has to adjust itself to shifting interests and changing topics. The use of machine learning techniques makes it possible to deal with this dynamic environment without placing any burden on the user.

All information filtering systems that automatically learn user interest profiles have a large time constant, which means it takes a long time before they start to make useful predictions. Because learning in natural language usually involves big search spaces, many examples are needed. Users will become disappointed with the initial bad performance, and might even stop using the system.

When we humans try to predict what somebody else would find interesting to read, we don't only look at what he read before, but we also use other knowledge available about this person. A logical answer to the problems would therefore be: use other information

available about a user's interests to improve the learning accuracy.

2 Related work

Malone describes three forms of filtering, *content based* filtering, *collaborative* filtering and *economic* filtering [Malone, 1987]. Content based information filtering looks at the words in the documents to find interesting ones. The words from previous documents the user has found interesting are used to find other interesting documents. Collaborative information filtering finds interesting documents by looking at judgements other people have given to documents. Judgements of people that have proven to be "similar" in the past will be weighed more heavily than others. Economic filtering has not often been used in text related tasks.

Content based information filtering is closely related to information retrieval and it is therefore not surprising that one of the dominant models in information retrieval, the *vector space model* ([Salton, 1983]), is also popular in information filtering.

Information filtering systems based on the vector space model using simple measures of word frequencies have been described in [Lang, 1995] and [Mukhopadhyay, 1996]. Recently, a technique called *Latent Semantic Indexing* (LSI) has been proposed as an alternative to the simple word based measures in information filtering [Foltz, 1992]. LSI uses a singular value decomposition of a word by document matrix to find latent semantic structures. Other techniques that are closely related to LSI have been proposed in [Hull, 1994] and [Oard, 1995].

The vector space model was originally designed for an information retrieval setting where a user has a one time goal on one specific topic. This model has difficulties handling the dynamic environment of information filtering where a user is interested in more than one topic and has changing interests. Other models that are used and that claim to solve these problems are Artificial Neural Networks ([Jennings, 1992] , [Schütze, 1995]) and Genetic Algorithms ([Sheth, 1993] , [Mock, 1994])

Most of these systems focus on filtering USENET net news articles, e-mails or technical reports for people in research environments because that makes it possible to quickly gather training data. In some sense, this has made life easier for the filtering systems, because the interests of people in these domains are usually well defined. Research with other "real world" applications could bring other desirable qualities of information filtering systems to light.

3 Problem Setting

This paper studies the use of machine learning for information filtering. Two typical examples of filtering systems were used to ground our study. *NewsWeeder* is a system to filter USENET net news articles ([Lang, 1995]). A user can give an article a rating from 1 to 5, from "essential" to "never want to see it again". For the experiments here, articles with ratings 1 and 2 formed the "interesting" class and articles with ratings 3 to 5 the "uninteresting" class. *Clea* is a system that finds the seminar announcements posted on the bulletin boards and presents them to the user. The user can indicate three things: he intends to go to this seminar, he finds it interesting, or he ignores it.

These systems try to present the user with news articles or seminars that he is most likely to find of interest. Like other information filters that attempt to learn the user's interests based only on observed user decisions, these systems will initially produce random recommendations and will require substantial training data before learning a successful user profile. We consider several additional sources of information that might be used to reduce this delay.

- The user's preferences on the World Wide Web were used. Indicators for these preferences are the pages in the user's bookmarks and the pages linked to the user's homepage.
- The users were asked to fill in a simple questionnaire of eight questions (see figure 1). Descriptions

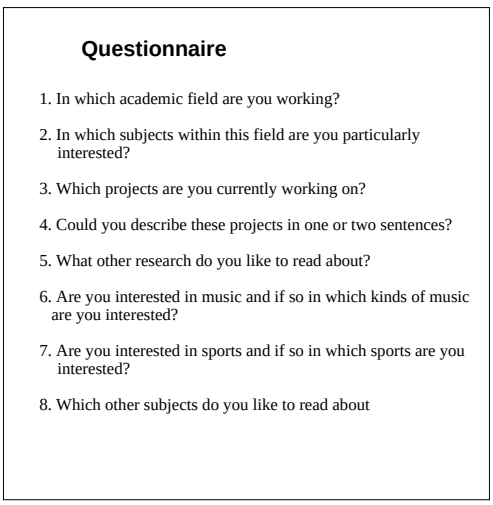

Figure 1: Questionnaire given to the users

of several sentences are preferred over single word descriptions, because many words are needed to use a vector space representation successfully. The answers were still rather brief, which is why the answers were used to retrieve documents from an unseen set of documents from the application under consideration, which were also used.

- For one of the users in our experiments datasets of both NewsWeeder and Clea were available. For this user we used positive examples from NewsWeeder to predict interests in Clea and vice versa.

The question this paper will try to answer is whether it is possible for an information filter to learn from these other sources of information. If this can be done, an information filter could be trained off-line and start working for a user with an acceptable performance from the first day of use.

4 Rocchio learning

The learning algorithm we used for our experiments is based on the relevance feedback method introduced by Rocchio [Rocchio, 1971] for the vector space retrieval model. It is a nearest class learning method with class prototype vectors using the $TFIDF$ (Term Frequency - Inverse Document Frequency) word weighting heuristic. This heuristic is based on two measures, the *term frequency* of word w, $TF(w)$, is the number of times w occurs in the document. The *document frequency* of w, $DF(w)$, is the number of documents in which the word occurs. There are many variations on the $TFIDF$ formula. For our experiments we used:

$$TFIDF(w) = TF(w) * log(\frac{|D|}{DF(w)}) \qquad (1)$$

where $|D|$ is the total number of documents in the document set.

A number of words with a high $TFIDF$-value are chosen to represent the document. These words and their values form a $TFDIF$-vector. A prototype vector for the class of interesting documents is learned by averaging $TFIDF$-vectors of previously seen documents classified to be interesting. The similarity between the vector of a new document and this prototype vector is calculated using the *cosine similarity* measure, which is the dot-product of these two vectors after they have been normalized to unit length. This cosine similarity is the relevance score of the document and can be used to rank documents.

Research in information retrieval has shown that these simple $TFIDF$-vectors based on frequency counts of words, often work as well as other text-representations. A benefit of this method is that it needs little preprocessing of the data, which is especially important in information filtering where large streams of incoming data have to be filtered.

In our initial experiments we averaged the $TFIDF$-vectors of all extra information available into one prototype vector. For every document, two similarities could now be calculated. One is the cosine similarity to the

vector learned by the Rocchio learner and the other is the cosine similarity to the prototype vector of the extra information. The relevance score of the document was defined as a weighted combination of these two similarities. Learning to rank documents using this relevance measure didn't give any improvement in learning accuracy.

5 Autoclass

In our opinion, the reason for the above mentioned poor performance in the first experiment was that the class of "interesting" documents in fact consists of various subclasses. These subclasses correspond to the different interests a user has. Averaging the vectors from all documents could reduce too much the weights of words that are important for only one or a few of these subclasses. To find out if this was the case, a second experiment was conducted in which the documents were clustered into subclasses. A separate $TFIDF$-vector for each subclass was created.

In order to accomplish this clustering, *Autoclass* was used [Cheeseman, 1990]. Autoclass uses a Bayesian approach to automatically cluster objects described by a vector of features. Neither the number of classes nor their description have to be known.

Most clustering algorithms search the space of possible clusterings of the documents and use a fixed model to calculate the quality of the clusters. The search ends when an acceptable clustering is found. Autoclass instead considers a space of different models and parameter settings. This model space is searched to find an acceptable set of class descriptions, using an Expectation Maximization (EM) algorithm. The best set is defined by a trade-off between the predictive accuracy over the documents and the complexity of the class descriptions. Prior expectations are used to favor simple class descriptions and thus try to keep Autoclass from overfitting the data. The subclasses formed are "fuzzy" in that documents have a membership value for each class.

In the model space Autoclass searches through, every feature is associated with a probability model. Autoclass requires these models to be chosen beforehand from a number of different possibilities. In our experiments, the $TFIDF$-weights of words were modeled as real valued features with a conditionally independent Gaussian normal distribution.

With this algorithm as a tool, the subclasses in the available information could be found. For any of these subclasses a seperate $TFIDF$-vector could then be created.

- The pages in a user's bookmarks and the pages linked to the user's homepage were clustered separately. Most homepages are organized by headers or the links are grouped in lists. Experiments were done to find the subclasses of webpages using this structure. This led to slightly worse results than using clustering.
- For the questionnaire, every answer formed it's own subclass. Unseen documents were clustered and only the clusters with the highest cosine similarity to the answers were used as an extra source.
- For NewsWeeder, the positive examples from every newsgroup formed a separate subclass. For Clea, the positive examples were clustered and a vector was created per cluster.

6 The final algorithm

Figure 2 shows the algorithm used to predict a relevance score for a document. This relevance score is obtained by combining the scores the document gets from a number of predictors. The Rocchio learner (see section 4), which is the only predictor that learns online during use of the system, is one predictor and the different sources used are other predictors.

The Rocchio learner has one prototype vector. The cosine similarity between the document's $TFIDF$-vector and this prototype vector is the relevance score of this learner

All the other predictors have a number of vectors, one per classified subclass. The cosine similarity between

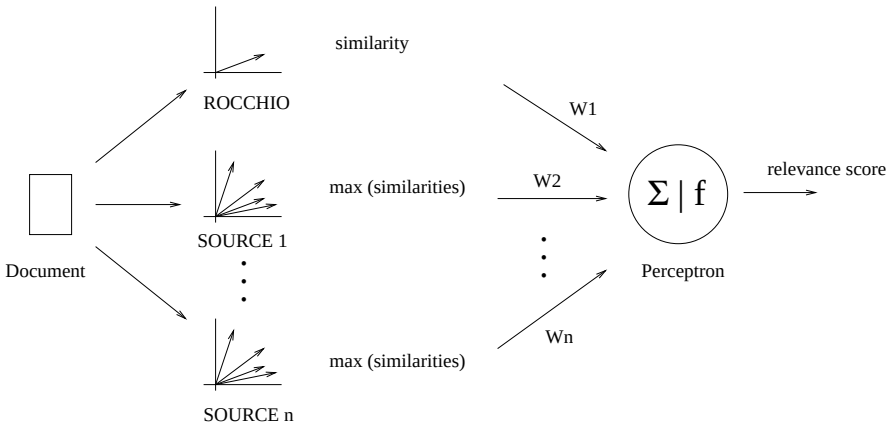

Figure 2: Algorithm for incorporating extra information

the document's vector and all these vectors is calculated. Similarities below a certain threshold are set to zero. The maximum of the remaining similarities is the relevance score given by this predictor. An alternative solution would be to calculate a combination of all similarities. This didn't work as well, which indicated that different vectors do represent different interests. Often only one or a few of the vectors have a high similarity with the document, i.e. the vectors representing the same topic.

In information filtering, giving a classification of documents into two classes "interesting" and "not interesting" is a somewhat arbitrary process. Only the "interesting" class is well defined. The "not interesting" consists of everything but the interesting documents. The transition between both classes is vague. The usual approach in information filtering is therefore to rank documents according to their relevance scores. Only the top of the ranking is presented to the user.

In order to combine the scores of the different predictors into a final score for the document, a simple perceptron was used. A perceptron is a threshold unit with a number of inputs x_i. Each input is multiplied by an associated weight w_i. The output of the perceptron is the result of an activation function f. In our experiments we used a linear activation function.

The perceptron is used to predict the relative position of the document in a ranking of documents. The relative position of a document is the position of this document in the ranked list divided by the length of this list. The output of the perceptron is a number between zero and one. Positive documents should get a relative position close to one, corresponding to the top of the list, and negative documents should get a lower relative position.

For training the perceptron, the target output for a positive document was 1, which corresponds to a perfect ranking of this document. The target output of a negative document was 0. The perceptron was trained after each example. It was not possible to iterate the weight updating process until the trainingset was ranked perfectly. With one perceptron and only up to six predictors it is not possible to get a perfect ranking. An upper-bound on the number of iterations was therefore set to 20.

7 Experiments and results

For our experiments data for four Clea users and one NewsWeeder user was available. The four Clea users had on average 180 examples, of which on average 17% were classified interesting. In NewsWeeder learning to rank the articles from all newsgroups in one ranking did not give good results. The experiments presented here were run separately for the two biggest newsgroups. This gave much better results because the weights of the perceptron could be learned independently for every newsgroup. The first newsgroup had 695 examples with 7% positives and the second newsgroup 226 examples with 23% positives.

The measure of evaluation of the results was the percentage of interesting articles found in the top 10% of the ranking. The default value for this measure is obtained when all documents are given the same score. The default performance is then equal to the percentage of interesting articles in that dataset.

The question we would like to answer is how much can we improve learning accuracy, and what is the maximum accuracy that can be achieved?

Figure 3 shows the results with the Clea dataset of user 1, for whom the least improvement in learning accuracy of the four users was measured. Figure 4 shows the results for user 4 with a much higher improvement of around 20%. The maximum achievable learning accuracy per person varied from 30% to 60%. This variation depended not only on the default for the user. For user 4 a high a high accuracy was achieved, despite a low default accuracy of 11%. Some user's interests seemed to be more well defined than others.

Figures 5 and 6 show the results on the NewsWeeder data. For these newsgroups, an improvement in learning accuracy between 20% and 30% was achieved. The most striking result here was that the Rocchio learner by itself was not able to do very much better than the default. A reason for this could be that both of these newsgroups had a wide variety of articles, which can be interesting to the user for many different reasons.Unfortunately, none of the more well defined newsgroups had enough data to do meaningful experiments with.

Another interesting question is if the perceptron was able to decide between useful and not useful sources. Figure 7 plots the development of the weights of the perceptron against the number of trainingexamples seen. This figure shows that our algorithm was able to find sources that are better than the others. The learning rate of the perceptron had to be set carefully in order to be able to disciminate between sources but on the other hand not to overfit the data. After enough trainingexamples were seen, the Rocchio learner usually seemed to be the best one. From the other sources, none seemed to be always better than the others.

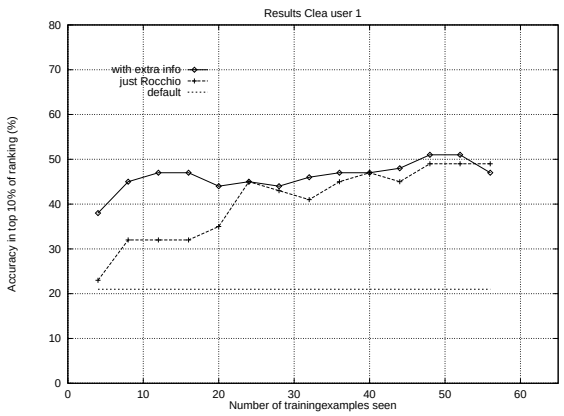

Figure 3: Results for Clea, average for User 1

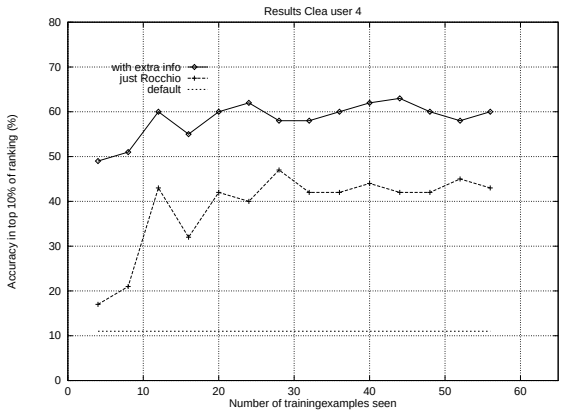

Figure 4: Results for Clea, average for User 4

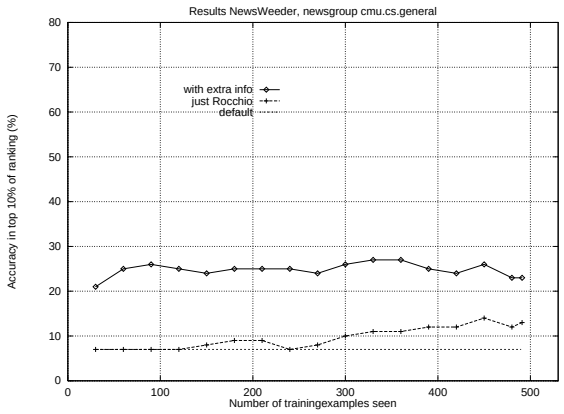

Figure 5: Results for NewsWeeder, average for first newsgroup

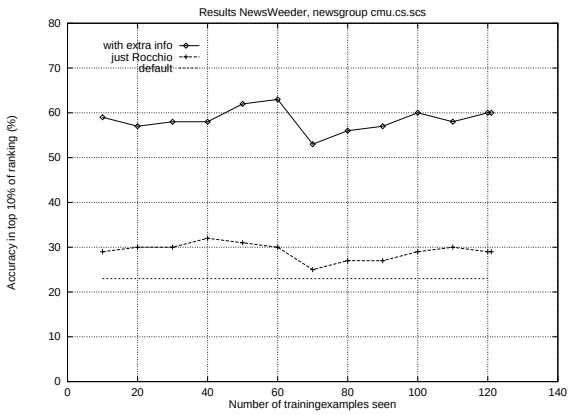

Figure 6: Results for NewsWeeder, average for second newsgroup

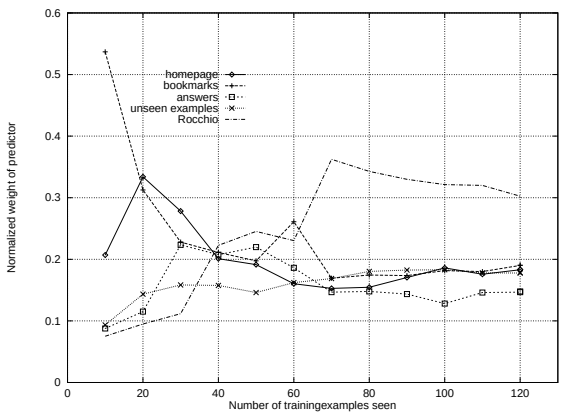

Figure 7: Results for NewsWeeder, perceptron weights for second newsgroup

8 Conclusion

In our experiments we have used three different sources available about a user's preferences: the pages he likes to visit on the World Wide Web, answers to a questionnaire and the model learned by another information filtering agent. Using all this information gave an improvement in learning accuracy over just using Rocchio learning. Typically this improvement was between 5% and 20% for Clea and 20% to 30% for NewsWeeder.

It is important that subclasses within the information of these sources are found. This is because most information filtering applications will only have to deal with a subset of the user's interests. Not splitting the information up in subclasses can hurt performance. Autoclass has proven to be an algorithm able to automatically classify documents into useful subclasses.

A surprising result was that in many cases, it did not only give improvements in the beginning, but it continued to do so, even after many training examples. This phenomenon seemed to appear in situations where articles about many different subjects fell in the class of "interesting" documents. In this case, our algorithm's better distinction between different interests seems to pay off best.

No conclusion can be drawn about which source of information works best to predict interests. This varied from user to user and for one user even from newsgroup to newsgroup. In these experiments, our algorithm has proven to be able to combine the sources in a useful way. Of course, not all extra sources used in this paper will normally be available, but some information, especially a questionnaire about a user's interests and the user's bookmarks or homepage, should be available in many situations.

9 Acknowledgements

We would like to thank Sean Slattery and Ken Lang for the datasets and parts of the code used in the experiments. This research is supported by ARPA under

grant number F33615-93-1-1330.

References

[Cheeseman, 1990] P. Cheeseman, J. Stutz and R. Hanson *Bayesian Classification Theory*, Tech. Rep. FIA-90-12-7-01, 1990

[Foltz, 1992] P.W. Foltz and S.T. Dumais *Personalized information delivery: An analysis of information filtering methods.* Communications of the ACM , VOL 35, 1992

[Hull, 1994] D. Hull *Improving Text Retrieval for the Routing problem using Latent Semantic Indexing.* Proceedings of the ACM SIGIR, 1994

[Jennings, 1992] A. Jennings and H.Higuchi *A Personal News Service based on a User Model Neural Network.* IEICE Transactions on Information and Systems, March, 1992

[Lang, 1995] K. Lang *NewsWeeder: Learning to Filter Netnews.* Proceedings of Machine Learning , 1995

[Malone, 1987] T.W. Malone, K.R. Grant, F.A. Turbak, S.A. Brobst and M.D. Cohen, *Intelligent Information Sharing Systems* , Communications of the ACM, 30 (5) , 1987

[Mock, 1994] K.J. Mock and V.R. Vemuri *Adaptive User Models for Intelligent Information Filtering* Proceedings of the Third Golden West International Conference on Intelligent Systems. Las Vegas, Nevada, 1994

[Menczer, 1995] F. Menczer and R.K. Belew *Artificial Live Applied to Adaptive Information Agents.* Spring Symposium on Information Gathering From Heterogeneous, Distributed Databases, 1995

[Mukhopadhyay, 1996] S. Mukhopadhyay, J. Mostafa, M. Palakal, W. Lam, L. Xue and A. Hudli *An Adaptive Multi-level Information Filtering System.* Proceedings of The Fifth International Conference on User Modeling, Hawaii, 1996

[Oard, 1995] D.W. Oard and N. DeClaris *Experimental Insvestigation of High Performance Cognitive and Interactive Text Filtering.* IEEE Conference on Systems, Man and Cybernetics, Vancouver, 1995

[Rocchio, 1971] J. Rocchio. *"Relevance Feedback in Information Retrieval"*, in The SMART Retrieval System: Experiments in Automatic Document Processing, Chapter 14, pages 313-323, Prentice-Hall Inc., 1971.

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

[Schütze, 1995] H. Schütze, D. Hull, J. Pedersen, *A Comparison of Classifiers and Document Representations for the Routing Problem*, International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995

[Sheth, 1993] B. Sheth and P. Maes *Evolving Agents for Personalized Information Filtering.* Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications, 1993