NewsWeeder: Learning to Filter Netnews

(To appear in ML 95)

Ken Lang
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
akl+@cs.cmu.edu

Abstract

A significant problem in many information filtering systems is the dependence on the user for the creation and maintenance of a *user profile*, which describes the user's interests. NewsWeeder is a netnews-filtering system that addresses this problem by letting the user rate his or her interest level for each article being read (1-5), and then *learning* a user profile based on these ratings. This paper describes how NewsWeeder accomplishes this task, and examines the alternative learning methods used. The results show that a learning algorithm based on the Minimum Description Length (MDL) principle was able to raise the percentage of interesting articles to be shown to users from 14% to 52% on average. Further, this performance significantly outperformed (by 21%) one of the most successful techniques in Information Retrieval (IR), term-frequency/inverse-document-frequency (tf-idf) weighting.

1 INTRODUCTION

As the number of participants in Usenet continues to multiply, so does the spectrum of topics covered. Users find it increasingly difficult to locate useful or interesting information as this diversity expands. There is a tradeoff each user must make between searching through fewer articles, and finding more information of personal interest. Ideally, a newsreader should allow the user to selectively choose this tradeoff such that increasing the scope of the search further would cause the marginal probability of additional articles being interesting to fall below a personal threshold. In reality, the tradeoff is usually made through subscribing to a small set of newsgroups and the use of simple keyword-matching profiles designed by each user.

Some research systems [Rewari, 92] have explored how to extend this idea by allowing more complex, rule-based, keyword-matching profiles to be designed by the user, but the average netnews reader will probably not be willing or able to build sufficiently complex ones. Consequently, first efforts have been made to have the profiles be automatically learned [Fischer, 91][Sheth, 94]. These systems all focus on looking at the article text to determine its relevance, which is known as *content-based filtering.*

Other systems [Resnick, 94][Goldberg, 92] have instead focused on using ratings from early readers of an article to predict later readers' ratings. This is known as *collaborative filtering.* The system described in this paper, NewsWeeder, uses both content-based and collaborative filtering. However, sufficient data on multiple users is not yet available, so the focus here will be on how the content-based part of the profile is learned.

NewsWeeder currently operates by the following procedure. When the user wants to read netnews, he follows a link to NewsWeeder's World Wide Web interface. There he chooses a topic, or newsgroup, of netnews articles he wishes to read, as is typically done in most newsreaders. In addition to using the traditional newsgroup hierarchy, the user can also use NewsWeeder's *virtual newsgroups*. For example, user Bob might go to the virtual newsgroup *nw.top50.bob* to see NewsWeeder's personalized list of the top 50 out of all articles, according to learned preferences for Bob. He is then presented with a list of one-line article summaries, sorted by predicted rating. The user selects a group of articles from these summaries and reads them sequentially. After each article is read, the user clicks on a rating from one to five, which replaces the need to indicate "next article" required by most newsreaders.

NewsWeeder collects the user's ratings for *active feedback* on the user's interests. This is preferable for a research system over *passive feedback* methods, such as measuring time spent reading, because performance can be more accurately evaluated. Further, the training signal

is probably stronger with active feedback, so better performance is likely. The drawback to active feedback is the extra effort the user must spend to decide on and click a rating, although with practice this effort appears to be acceptably minimal.

Each night, the system uses the collected rating information to learn a new model of the user's interests. The rating data is also collected for off-line learning experiments, which gives good evidence for on-line performance and comparisons between learning methods. These off-line experiments are the subject of this paper.

2 APPROACH

The experiments performed with NewsWeeder cover a small set of points in the large space of possible filtering methods. We will describe the space as a map of choices to be made in designing a text filter. We also describe the choices made in NewsWeeder and the reasoning behind them.

2.1 REPRESENTATION

In NewsWeeder, raw text is first parsed into generalized words, called *tokens*. Tokens include punctuation and other specialized symbols that have been engineered out of the structure found in the article headers. For example, in addition to typical words such as "seminar" counting as tokens, the punctuation mark "$" and the symbol "Newsgroup:comp.ai" are also tokens. Using noun phrases as tokens has also proven [Evans, 91] to be useful. However, we have kept the granularity of feature size down to the word level to avoid the linguistic knowledge that using noun phrases and larger structures usually entail building.

Next, NewsWeeder creates a vector of token counts for the document. This vector is the size of the total vocabulary with zeros for tokens not occurring in the document. Using this type of vector is sometimes called the *bag-of-words* model, and is the basis for our representation. While the bag-of-words model does not capture the order of the tokens in the document, which is necessary for linguistic or syntactic analysis, we assume it captures most of the information needed for filtering purposes.

As a next step, it is common in IR systems to group the tokens together by their common linguistic roots, a procedure called *stemming*. In NewsWeeder, the tokens are left in their unstemmed form. While better performance could probably be achieved by stemming in the short-term due to the larger statistical samples it creates, the long-term research goal is to not waste the extra information contained in the use of exact tokens. We plan to eventually try to use this extra information by looking at larger-size samples of unrated text to get more reliable statistics on rare words. This technique would attempt to use the large amounts of available unrated text in an unsupervised learning approach to bias the supervised learning on the smaller set of rated articles. Further, there is evidence [Yang, 94] that stemming can hurt performance if the approach used is able to make strong enough statistical inferences about the unstemmed tokens.

2.2 LEARNING METHODS

Once the text has been processed into a standardized vector, the choice of which learning method to apply must be made. Given that this vector is on the order of 20,000-100,000 tokens long, care must be taken to avoid the pitfalls of learning in high-dimensional input spaces. The "curse of dimensionality" makes learning difficult because many more examples are needed to determine which dimensions are important in a space in which all examples begin to look equidistant.

One way to combat the high-dimensional space is to reduce its dimensionality before applying further techniques. For example, Singular Value Decomposition (SVD) [Dumais, 88] or auto-encoding neural nets attempt to reduce the size of the space while maximally retaining the information contained in the original representation. The initial NewsWeeder experiments have been done with no dimensionality-reduction techniques other than throwing out words deemed less-useful, according to a heuristic described later. Earlier experiments [Lang, 94] with the task of predicting newsgroup from the text body indicated that techniques that used SVD were outperformed by techniques without such transformation. This result was probably caused by a loss of information when using the transformation.

Of the many possible learning methods available, two were chosen to fit the characteristics of the domain. The first is a well-tested, popular technique from IR called term-frequency/inverse-document frequency weighting (tf-idf) [Salton, 91]. The Stanford Netnews Filtering Service [Yan, 92] relies on this technique, although their efforts are focused on solving delivery and indexing problems for large numbers of users, rather than trying to develop better learning models. Tf-idf provides a well-tested benchmark from which other learning models can be compared.

2.3 TF-IDF WEIGHTING

The assumptions behind tf-idf are based on two empirical observations regarding text. First, the more times a token t appears in a document d (called the *term frequency*, or $tf_{t,d}$), the more likely it is that t is relevant to the topic of d. Second, the more times t occurs throughout all documents (called the *document frequency* or df_t), the more poorly t discriminates between documents. For a given document, these two terms are combined into weights by multiplying the *tf* by the inverse of the *df* for each token. Often the logarithm of *tf* or idf are taken in order to de-emphasize the increases in weight for larger values. In experiments with NewsWeeder, the weight used for token t in document d was

$$w(t,d) = tf_{t,d} \log(|N|/df_t)$$

where N is the entire set of documents[1]. The way in which tf-idf vectors are compared also takes advantage of the domain. Because documents usually contain only a small fraction of the total vocabulary, the significance of a word appearing is much greater than of it not appearing. To emphasize the stronger information content in a word appearing, the cosine of the angle between vectors is used to measure the similarity between them. The effect of this metric can be seen in the following example. Suppose two documents each contain a single word, but the words are different. The similarity of the documents would then be zero, since the cosine of the angle between the two perpendicular vectors is zero. A more unbiased learning technique that did not take advantage of this domain feature would usually group the two documents as being very similar, since all but two of the elements in the lengthy vectors agreed (i.e., they were zero).

Using tf-idf and the cosine similarity metric, there are many ways to then classify documents into categories. For example, any of the family of nearest neighbor techniques could be used. In NewsWeeder, the documents in each category are converted into tf-idf vectors, normalized to unit length, and then the average is taken to get a prototype vector for the category. The advantages to doing this are speed of computation and more compact representation. To classify a new document, the document is compared with each prototype vector and given a predicted rating based on the cosine similarities to each category of rating. In this last step we convert the results from a categorization procedure to a continuos value using a linear regression, which we will describe in more detail subsequently. While we could have simply lumped all the highly-rated categories into one and used a single similarity measure to provide the rating prediction, we felt this would not fully take advantage of the gradations of relevance feedback obtained from the user.

2.4 MINIMUM DESCRIPTION LENGTH (MDL)

The alternative machine learning technique we compare to tf-idf weighting is based on the MDL principle [Rissanen, 78]. The MDL principle provides an information-theoretic framework for balancing the tradeoff between model complexity and training error. In NewsWeeder's domain, this tradeoff involves how to weight each token's importance and how to decide which tokens should be left out of the model for not having enough discriminatory power. To derive the MDL principle, we begin with Bayes' Rule

$$p(H|D) = \frac{p(D|H)\,p(H)}{p(D)}$$

We wish find the hypothesis H that maximizes p(H|D), the probability of H given the observed data D. By Bayes' Rule, this is equivalent to maximizing p(D|H)p(H)/p(D). Since p(D) doesn't depend on H, we can just maximize p(D|H)p(H); equivalently, we can minimize

$$-\log(p(D|H)) - \log(p(H)).$$

From information theory [Shannon, 48], we know that $-\log(p(X))$ (base 2) is equal to the size in bits of encoding event X in an optimal binary code. So, the MDL interpretation of the above expression is that, to find the most probable hypothesis given the data, we should find the hypothesis which minimizes the total encoding length. This encoding length is equal to the number of bits required to encode the hypothesis, plus the bits required to encode the data given the hypothesis. The MDL principle thus expresses the intuitive notion that finding a good model involves finding the right balance between simpler models (which take fewer bits to describe than more complex models) and models that produce smaller error when explaining the observed data.

The essential difference between the MDL approach and the popular Maximum Likelihood (ML) method is that ML treats the cost of encoding all models as equal. That is, the prior probability of all hypotheses is uniform, and so can drop out of the optimization.

2.4.1 MDL Applied to NewsWeeder

Similar to the tf-idf approach, we will first perform a rating-categorization step, and then convert the categorization similarities into a continuous rating prediction. Given a document d with token vector T_d (containing l_d non-zero entries[2]) and training data D_{train}, the most probable category c_i for d is that which minimizes the bits needed to encode T_d plus c_i

$$\arg\max_{c_i}\{p(c_i|T_d, l_d, D_{train})\} =$$

$$\arg\min_{c_i}\{-\log(p(T_d|c_i, l_d, D_{train})) - \log(p(c_i|l_d, D_{train}))\}$$

For simplicity, and because we are not presently focused on taking advantage of document length as a predictive feature, we treat the last term as constant. We now must carefully construct a model for the probability distribution of T_d.

2.4.2 A Probabilistic Model for Token Occurrences

In choosing a probabilistic model for token distributions, we want to find the most descriptive function that takes

[1] Using the log of the tf was also tried, but was found to decrease performance slightly.

[2] We use l_d as a measure of document length, redefined to mean "number of unique tokens in the document".

advantage of what we know about the problem space, and with the fewest number of parameters. Pragmatically, we also want it to be fast, since it will be used over large amounts of data.

A common assumption in IR is that the probabilities that two words occur in a document are independent. This assumption is not particularly well-grounded empirically, but it reduces the complexity of the problem space significantly enough that it should be tried before anything else. Our approach uses this assumption, but also allows for a degree of dependence on document length for each word's probability to range from highly dependent to completely independent. Since we are treating special tokens as just additional words in the model, word probabilities may or may not depend on the document length. For example, the token "Re" used as an abbreviation in the subject line for "Regarding", can appear *more* frequently in shorter documents than longer ones, since they are often one-line responses quoting a short section of a longer article. More typically, an ordinary word such as "flaming" appears with greater frequency the longer the document. Formally, our independence assumption is that the probability of the data in a document given its length and category is the product of the individual token probabilities

$$p(T_d|c_i, l_d, D_{train}) = \prod_i p(t_{i,d}|c_i, l_d, D_{train})$$

where $t_{i,d}$ is a binary value indicating whether or not the token i occurred at least once in document d.

Now, we wish to derive a probability estimate for $t_{i,d}$, but we want to avoid a computationally expensive optimization step for the parameters of whatever model we decide on. One way to do this is to compute the following additional statistics from our training data, and use them as the parameters in the model:

$t_i = \sum_{j \in N} t_{i,j}$: the number of documents containing token i

$r_{i,l}$: a correlation estimate[3] [0-1] between $t_{i,d}$ and l_d

Each statistic is computed for each category, and for the total across all categories. The objective is to establish a general "background" distribution for each token, and a category-specific distribution. We will use the subscript $[c_k]$ to designate that in modeling the category-specific case, an equation is dependent only on the statistics from a particular category of articles. However, the equation is equally valid for modeling the background distribution by removing the $[c_k]$ and using the statistics applied to all of the articles. Following this convention, if the token distribution is a simple binomial, independent of document length, we havc

$$p(t_{i,d} = 0|[c_k]) = 1 - t_{i[,c_k]} / |N_{[c_k]}|$$

However, if the token probability is dependent on document length, we can use the approximation

$$p(t_{i,d} = 0|l_d[,c_k]) = \left(1 - t_{i[,c_k]} \Big/ \sum_{j \in N_{[c_k]}} l_j\right)^{l_d}$$

The above two distributions can then be combined in a mixture model by weighting them with $r_{i,l}$

$$p(t_{i,d} = 0|l_d[,c_k]) = \left(1 - t_{i[,c_k]} / |N_{[c_k]}|\right)^{(1-r_{i,l})} \times \left(1 - t_{i[,c_k]} \Big/ \sum_{j \in N_{[c_k]}} l_j\right)^{l_d \times r_{i,l}}$$

We now have a choice to make between two probability distributions for each token: the category-specific, or general distribution. We hypothesize that each token either truly has a specialized distribution for a category, or that the token is unrelated to that category and just exhibits random background fluctuations. The MDL criteria for making the decision between these hypotheses is to choose the category-specific hypothesis if the total bits saved in using this hypothesis

$$\sum_{d \in N_{c_k}} -\log(p(t_{i,d}|l_d)) - [-\log(p(t_{i,d}|l_d, c_k))]$$

is greater than the complexity cost of including the extra, category-specific parameters. At present we use a small constant[4] to represent this complexity cost.

An additional pragmatic advantage to this probabilistic model choice is that when the logs are taken of the probabilities to get costs in bits, the probability calculation for each article's words becomes a simple, linear one that can be computed in $O(l_d)$ rather than the much longer $O(|dictionary|)$[5]. This is due to the ability to precompute the sum of the bits required to encode no words occurring. From this sum the bits required for an actual document can quickly be computed.

We can now compute the similarity of a given document to each rating category. The similarity is inversely proportional to the number of bits required to encode T_d

[3] Properly, r should be the standard statistical sample correlation between a token's occurrence and the document length, bounded below by 0. In these experiments the fraction [0-1] of documents not containing token t was used as a proxy for this estimate, since we were mostly concerned with inaccurate probabilities for the most frequent tokens.

[4] More theoretically-correct complexity costs were tried, but none worked as well as using the simple constant = 0.1

[5] This pragmatic advantage is also why a multiplicative mixture model (which becomes linear after taking the log) was used instead of the more typical linear model.

using the combination of probability distributions described previously.

2.5 LEARNING ALGORITHM SUMMARY

Now we will make explicit the entire algorithm used in both the tf-idf and MDL experiments:

1. Divide the articles into training and test (unseen) sets.
2. Parse the training articles, throwing out tokens occurring less than three times total.
3. For tf-idf, also throw out the M most frequent tokens over the entire training set.
4. Compute t_i and $r_{i,l}$ for each token.
5. For tf-idf, compute the term weights, normalize the weight vector for each article, and find the average of the vectors for each rating category.
6. For MDL, decide for each token t and category c whether to use $p(t|l,c)=p(t|l)$, or to use a category-dependent model for when t occurs in c. Then pre-compute the encoding lengths for no tokens occurring for documents in each category.
7. For tf-idf, compute the similarity of each training document to each rating category prototype using the cosine similarity metric.
8. For MDL, compute the similarity of each training document to each rating category by taking the inverse of the number of bits needed to encode T_d under the category's probabilistic model.
9. Using the similarity measurements computed in steps 7 or 8 on the training data, compute a linear regression from rating category similarities to continuous rating predictions[6].
10. Apply the model obtained in steps 7-9 similarly to test articles.

We now have a continuous prediction of the user's ratings. This two-stage algorithm of text categorizing and then regression was motivated by three considerations. First, the categories were kept distinct instead of combined into relevant and non-relevant categories to fully take advantage of the user's feedback gradations. Second, we wanted our final prediction to be matched to the same absolute rating scale as the user's feedback to aid in the user's evaluation of the prediction, rather than just obtaining article ranking information. Third, we needed to combine all the similarity information somehow, and linear regression is one of the simplest ways possible.

[6] The standard least-squares regression was used for both the MDL and tf-idf experiments, but we pre-processed the input variables as follows. First, the similarities are normalized to range from 0-1. Second, after empirically observing that these normalized similarities needed more separation, we then used the cube of this normalized similarity as the inputs for the regressions. This is not as arbitrary as it may sound. If the token independence assumption were adhered to, we would expect to use a regression on the exponential of the log of the probabilities. However, because this assumption is wildly violated, a function less than exponential works well, although linear appears to be too weak. Powers from three to six all seem to work equally well.

2.6 EVIDENCE FUSION

The results presented in this paper are based solely on the content-based rating predictions. However, the NewsWeeder agent combines an element of collaborative filtering in actual on-line use. When a new article is prefetched into the system, and after a few readers have had a chance to rate the article, the prediction of the next reader's rating R is based on three components:

- the content-based rating prediction for R
- the ratings already given by other users
- the content-based rating predictions for other users

The above components are combined into the final prediction for a user by a simple weighted average. The components are listed in the order of weight size, from most important to least. These weights are non-adaptive and arbitrarily chosen, but subjectively have been observed to provide an interesting serendipitous sampling of articles scoring highly in other user's filters. One of our next research goals is to learn a model of how to fuse these rating evidence components more effectively by correlating the interests of multiple users.

3 RESULTS

There are several evaluation metrics that can be used to evaluate performance. We will look at one of the common evaluation metrics used in IR, *precision*. Precision is defined as the ratio of relevant documents retrieved to all documents retrieved, given some method for choosing a subset of documents to retrieve. Our method is to take the top 10% of highest-predicted-rating articles, motivated by the following reasoning. Current newsreading filter techniques are poor enough that many people opt not to read any articles at all. Thus, the challenge is to raise the signal-to-noise ratio high enough in a small set of filtered articles that the value exceeds the costs for these users.

Another way we will analyze the results is by looking at the confusion matrix of errors generated by the text classifier, to get a better understanding of where the method is going wrong.

3.1 DATA

The labels given to the user as the meanings of the ratings are provided in Table 1. For evaluation purposes, we define "interesting" as articles rated 2 or better, and define the numerical rating of "Skip" articles to be 4.5.

Table 1: Rating Labels

Rating	Label	Intended Use
1	Essential	For articles not to be missed if at all possible
2	Interesting	For articles of definite interest
3	Borderline	For articles the user is uncertain about his interest in and would rather not make a commitment
4	Boring	For articles not interesting
5	Gong	For articles the user wants to heavily weight against seeing again, perhaps because they are so clearly irritating to have in the list
Skip	Skip	For articles the user does not even want to read (note that this category may cover several of the above ratings, as they can only be used if the user actually requests to see the article)

While over 40 users have registered to use NewsWeeder, only two have stuck with it through its development process to have enough data for evaluation at present. In Table 2, we summarize the two data collections, which were recorded over a one-year period. Since our model assumes a stable distribution pool, and because it has no temporal dependence, users' interests that lasted less than the one-year period will add some amount of error to the performance[7]. Because different users have different styles of newsreading, it is difficult to make comparisons between the users' ratings. For example, the "Skip" category, which means that the user saw the one line article summary, and decided not to read the article, can be optionally used depending on the user's style. Typically, a user will pick out, read, and rate the articles desired from the list, and then may or may not choose to rate the remaining articles "skipped". For small, but important newsgroups the user may want to "clean up" the list of articles so that none are missed, but for larger newsgroups, just letting the articles expire naturally off the list may be preferable. Thus, even though User B has rated 16% of the articles as interesting (a rating of 1 or 2), it is possible for a considerably smaller percentage of interesting articles to be in the newsgroups read by User B.

[7] In fact, earlier experiments with an eight-month period of data had fewer training examples than the one-year period, but we observed approximately 10-20% *better* performance with this smaller data set. Probably this was due to a more stable set of interests.

Table 2: Article/Rating Data for Two Users

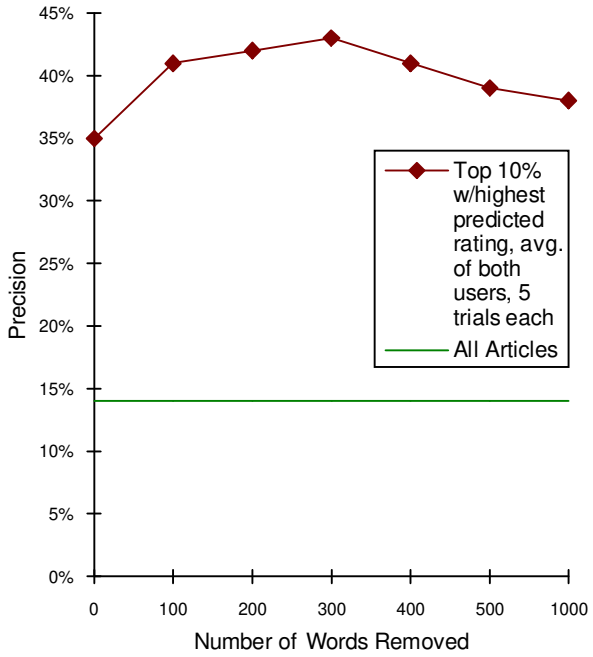
Rating	User A	User B
1	27 (1%)	29 (3%)
2	475 (11%)	143 (14%)
3	854 (20%)	67 (6%)
4	935 (22%)	56 (5%)
5	57 (1%)	17 (2%)
Skip	1995 (46%)	732 (70%)
Total	4343	1044
Total Interesting (1 or 2)	502 (12%)	172 (16%)

3.2 TF-IDF PERFORMANCE ANALYSIS

In using tf-idf, IR experts typically use what they call a "stop-list" of hand-chosen, content-free words that are automatically removed from the articles as insignificant. While this may seem like an arbitrary and difficult list to accurately create, it tends to improve results significantly enough to be worthwhile in typical IR text collections. Netnews, however, is a very dynamic environment in which the set of content-free words are constantly shifting, and so we assume that a static list would be a poor choice. The assumption in the tf-idf weighting scheme that very frequent words tend to be less predictive provides a means for automating this stop-list. In experiments with such an automated list, we found that indeed it did improve results significantly to have some of the most frequent words removed. However, in looking at the list of words removed, the words were occasionally clearly not content-free. This effect of throwing out some of the predictive words along with the less useful ones is clearly undesirable, but is better than keeping all of them. Graph 1 shows the effect of removing the top N most-frequent words on precision in the top 10%, highest-predicted-rating articles. Five trials were performed for each user with different training/test set splits (80%/20%) for each trial, and the results were averaged over both users. Removing around 100-400

words appears to be the range at which the stop-list is most successful, with the best precision of 43% being reached at 300 words removed.

Graph 1: Performance after Removing Most Frequent Words



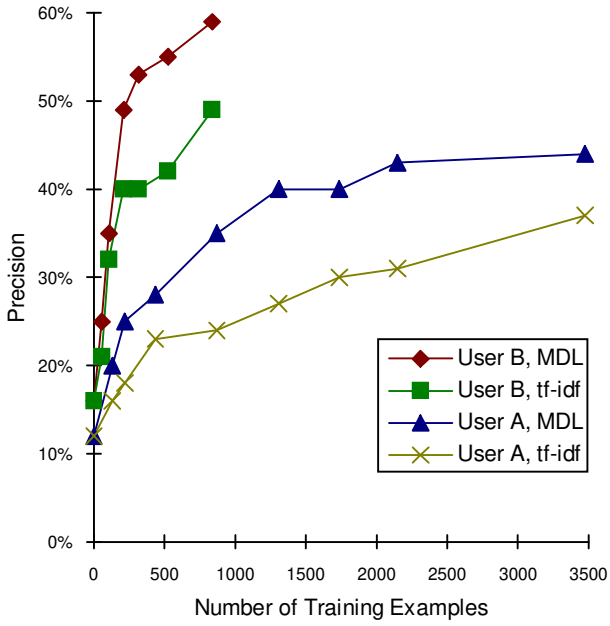
If this is indicative of real performance, it means the user could look forward to reading interesting articles at a rate more than three times higher in the filtered list than looking at all articles in his subscribed newsgroups. Of course this means the user will miss the other 59% of interesting articles (11% of the remaining 90%, filtered-out articles). The user who ordinarily wouldn't read any netnews at all due to low signal-to-noise would probably benefit significantly, though.

3.3 MDL PERFORMANCE ANALYSIS

The MDL approach described so far should not be affected positively by removing frequent words, and empirically this was found to be true. We show next in Graph 2 how well the MDL approach performed versus the tf-idf approach, for both users A and B over a varying training set size. Performance is again measured as the precision (percentage of interesting articles found in the top 10% with highest predicted rating) averaged over five trials with randomized training/test set splits.

The MDL learning algorithm eventually reached a precision of 44% for the top 10% of highest-predicted-rating articles for User A, and 59% for User B. This compares favorably with the tf-idf approach (37% and 49%), finding 21% more interesting articles for each user than tf-idf out of the same number of selected articles. User B's percentage of interesting articles was 16% overall (compared with 12% for User A), so it is not surprising to see better performance for User B than User A with far fewer examples.

Graph 2: Performance of Tf-idf vs. MDL on Test Set (Avg. of 5 Trials)



In Table 3, we see an example of the confusion matrix for MDL's categorization of articles for User A in a single trial. This matrix shows where the text classifier errors are found, before the linear regression is performed. These numbers reflect what the MDL filter would predict the user's rating to be if it simply chose the most likely rating category, rather than passing all of the categorization information through the linear regression. Note that it still tends to get a large percentage (65%) correct along the main diagonal. Further, the precision obtained by just using the articles predicted to have a rating of 1 or 2 in the first two columns agrees with our results after using the regression (28/64=44%). In general, the performance after the regression step tends to meet or exceed the precision obtained by this method of using only the categorization outputs.

Table 3: MDL Confusion Matrix for User A's Test Articles

(based on most-likely rating category)

		Predicted Rating						Total
		1	2	3	4	5	Skip	
Actual Rating	1	0	2	1	1	0	1	5
	2	2	24	23	16	1	29	95
	3	0	20	77	44	1	28	170
	4	0	2	26	104	1	54	187
	5	0	0	2	5	2	2	11
	Skip	0	14	19	4	1	361	399
Total		2	62	148	174	6	475	867

4 CONCLUSION

The data presented gives evidence for the following statements. First, machine learning and the MDL approach can have significant benefits in performance over one of the best techniques developed in Information Retrieval, tf-idf. Second, it is possible to gain some value from learning to filtering netnews for at least a subset of users. And third, it can be done within a reasonable number of training examples. We have only explored the potential of a portion of the available information that can be leveraged in the content-based filtering side, and not explored at all the collaborative filtering potential. The results leave us optimistic that expanding the use of the available information will permit automatic learning of useful news filtering profiles for a wide variety of users.

5 FUTURE WORK

Once more multi-user data has been collected, we hope to find out how much the content-based techniques will synergize with the collaborative techniques. Further, the MDL technique presently does not use the term frequency fully—only the knowledge of whether a term occurred at least once in an article. We hope that adding this information to make MDL and tf-idf on equal footing will widen the gap between their performance. Lastly, we hope to take advantage of large numbers of freely available, unrated news articles to bias the learning with stronger statistics on the global word distributions and dependencies.

Acknowledgments

I would like to thank my advisors, Tom Mitchell and Andy Witkin, for their advice and guidance through this research, and my colleagues Rich Caruana and Geoff Gordon for their comments and discussions on this work.

This research is based on work supported by grants from Digital Equipment Corporation and the Advanced Research Projects Agency.

References

[Dumais, 88] Susan Dumais et al. Using Latent Semantic Analysis to Improve Access to Textual Information. Proceedings of CHI-88

[Evans, 91] David Evans et al. A Summary of the CLARIT Project, Technical Report, Laboratory for Computational Linguistics, Carnegie-Mellon University, September 1991

[Fischer, 91] G. Fischer and C. Stevens. Information Access in Complex, Poorly Structured Information Spaces, *Human Factors in Computing Systems, CHI'91 Conference Proceedings*, ACM, New Orleans, LA, April 1991, pp. 63-70

[Goldberg, 92] D. Goldberg, et al. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35, 12 (1992), pp. 61-70

[Lang, 94] Ken Lang. Internal Research Report, Carnegie Mellon University, 1994, WWW:http://anther.learning.cs.cmu.edu/res_sum.ps

[Resnick, 94] Paul Resnick et al. *GroupLens: An Open Architecture for Collaboratie Filtering of Netnews*, Internal Research Report, MIT Center for Coordination Science, March 1994

[Rewari, 92] Anil Rewari. et al. AI Research and Appplications in Digital's Service Organization. *AI Magazine* :68-69, 1992

[Rissanen, 78] J. Rissanen. Modelling by Shortest Data Description, Automatica, 14:465-471, 1978

[Salton, 91] Gerard Salton. Developments in Automatic Text Retrieval, *Science*, 253:974-980, August 1991

[Shannon, 48] C. E. Shannon. A Mathematical theory of Communication, *Bell Sys. Tech. Journal,* 27: 379-423, 1948

[Sheth, 94] Beerud Sheth. *A Learning Approach to Personalized Information Filtering*, Master's Thesis, M.I.T., February 1994

[Wallace, 94] F. Mosteller and D. L. Wallace. *Applied Bayesian and Classical Inference: The Case of the Federalist Papers*, pp. 65-66, Springer-Verlag, New York, 1984

[Yan, 92] T.W.Yan and H. Garcia-Molina. Index Structures for Selective Dissemination of Information, Technical Report STAN-CS-92-1454, Stanford University, 1992

[Yang, 94] Yiming Yang. An Example-Based Mapping Method for Text Categorization and Retrieval, *ACM Transactions on Information Systems*, Vol. 12, No. 3, July 1994, pp. 252-277