ifile: An Application of Machine Learning to E-Mail Filtering*

Jason Rennie
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jr6b@andrew.cmu.edu

December 1, 1998

Abstract

With the proliferation of electronic mail in the modern era, it becomes ever more important to devise methods for the organization, categorization and searching of such mail. Mail filtering is such a method for the organization of incoming e-mail. In this paper, we critique modern mail filters and describe how Machine Learning techniques could improve upon them without eliminating the benefits they provide. We then present **ifile**, a mail filter which makes use of the text classification algorithm naive Bayes. Furthermore, we present experiments showing its classification performance on sets of private e-mail and point out some of the issues involved with creating a Machine Learning based mail filter. We conclude that **ifile** eliminates some of the problems with modern mail filters while still maintaining high classification accuracy.

1 Introduction

Most modern mail clients allow the user to organize his or her mail into what are called folders or mailboxes. This allows the user to group together mail messages which are similar to each other. Many people make use of this flexibility allowed by mail clients. However, making full use of this organizational benefit can be both time consuming and tedious without the help of an agent or program. In response to this, programs called "mail filters" have been created to automatically place incoming mail into folders according to pre-defined rules which the user can specify. Such mail filters do help to relieve the organizational dilemma, but they are still a long stretch from solving the problem.

Traditional mail filters have three major problems associated with them:

*ifile is freely available from http://www.cs.cmu.edu/~jr6b/ifile/

- Each user must determine her own rules for mail filtering. This can be time consuming and require the navigation of a complex interface.
- Any changes in the folder hierarchy may require significant changes in the set of filtering rules.
- Content-based filtering[1] is difficult.

And yet, Machine Learning methods have the potential to reduce or eliminate all of these problems without removing the highly accurate filtering which they provide.

In response to this, we have created mail filter, named **ifile**, which is based on a simple text classification algorithm known as naive Bayes. It is written for the mail client EXMH and makes use of user actions which are common for most mail clients. It thus eliminates much of the maintenance involved with most modern mail filters. Additionally, the nature of the naive Bayes algorithm allows for filtering which is based on the content of a document.

2 Approach

2.1 Modern Mail Filters

Today, many users of e-mail make use of mail filters to classify incoming e-mail. This allows a reduction in the amount of time required to organize one's e-mail corpus. Two e-mail filters which are representative of the current selection of filters are **procmail** and **slocal**. Both of these filters allow the user to concoct a set of rules by which incoming e-mail is classified. The expressive power of these rules is sizable. In the case of procmail, the user can match against regular expressions in any part of an e-mail message. However, to make use of such expressive power requires significant time commitment. The user must spend time both creating rules and updating them as her preferences change. And while one might imagine that such rules could be used to filter e-mail based on message content, it is unlikely that any user is willing to spend the time needed for the devising and encoding of such rules. Thus, currently available filters are well suited for users who are willing to spend time maintaining a set of rules and who classify their e-mail according to e-mail addresses and/or keywords found in e-mail headers. However, for those who wish to filter their e-mail according to content or do not have the time and understanding to devise and encode filtering rules, a different type of mail filter is needed.

2.2 Machine Learning

We would like to devise an accurate mail filter which eliminates the problems of modern mail filters. One way in which this might be possible is through the application of Machine Learning methods. Text classification algorithms have been shown to perform quite well and such an algorithm could eliminate or reduce the problems with modern-day mail filters.

[1] filtering which is based on the English meaning of the body of a document

A number of studies have been done which show Machine Learning techniques performing e-mail related classification tasks quite well. Ken Lang uses TF-IDF to score high classification accuracy on the task of classifying an array of netnews documents into newsgroups in which they belong [Lan95]. David Lewis and Kimberly Knowles use a TF-IDF classifier to successfully match e-mail messages with their "parents" [LK97]. William Cohen tests TF-IDF and a rule-based classifier on a corpus of her own e-mail [Coh96]. Finally, Thorsten Joachims uses Support Vector Machines to classify Reuters documents [Joa98]. Studies which concern themselves with the classification of e-mail documents generally show Machine Learning algorithms achieving accuracies of 80% or better, even when the classification task is across a wide range of classes. Thus, Machine Learning algorithms are quite likely to be capable of handling the task of filtering personal e-mail.

By using such algorithms, we can eliminate many of the problems with modern mail filters. An on-line algorithm would eliminate the need for the user to provide a set of rules to the mail filter. It should also be possible to have such an algorithm detect when the user has made significant changes in her classification preferences by observing its own performance. Furthermore, Machine Learning makes the task of content-based mail filtering feasible. Not only does Machine Learning have the potential to eliminate many of the problems of current mail filters, but there is already been strong evidence in the literature that such an algorithm could accurately classify personal e-mail.

2.3 On-Line Learning

In the development of a Machine Learning based mail filter, one has to consider a number of issues which are not normally tackled by those who evaluate and experiment with Machine Learning algorithms. We must take in to account the method by which features (words) are to be selected for the purpose of classifying e-mail messages. The choice of training examples must also be carefully made so as to maximize the amount of information the algorithm can use to base its decisions.

The majority of text classification algorithms make use of feature selection. In the context of text categorization, feature selection limits the number of words which are used for the purposes of classification. This can be used to increase the performance of an algorithm on certain data sets. It can also be useful for making the classification task less computationally daunting. [YP97] compares the performance of four different feature selection methods and determines that document frequency is a reasonable method. In order to make use of such a feature selection method in an on-line context, we create the notion of the *age* of a word and remove a word from our database when $\log_2 age - 1$ becomes greater than the document frequency of a word. The *age* of a word, w, is defined as the number of documents which have been filtered by the ifile system since w was first encountered. Using this metric, only words which occur relatively frequently are used for classification purposes. When ifile uses such feature selection, it commonly retains 2000-8000 features (words) for classification purposes.

In the context of EXMH, the mail client for which **ifile** was developed, there are only two instances in which the mail filter has access to information about e-mail messages. These are the *filtering* and *refiling*[2] actions. When a new message is incorporated, it is given to

[2]Other commonly used terms include moving and transferring

the mail filter to be classified in to one of the user's folders. After a piece of mail is filtered, the user has the capability of refiling mail from one mailbox to another. Considering these two opportunities for acquiring knowledge about the user's filtering preferences, we decided to use as a training example every e-mail which is filtered by the **ifile** system. When a piece of mail is filtered, **ifile** generates a class label and then adds the word statistics for that mail to the word statistics of the folder for which it was given the label. When a piece of mail is refiled, word statistics for that mail are moved from one folder to another. Using this method of labeling training examples, we maximize our set of examples while keeping a very good model of the user's filtering preferences. The only case where this method may do harm is when a large number of e-mail messages are filtered. If the labelings for the first few messages are wrong, this may cause later e-mail messages to be labeled incorrectly even though they may have been labeled correctly had a less aggressive method been used. This method makes the assumption that a user will refile all incorrectly labeled e-mail and will do so before the next batch of filtering is performed.

2.4 ifile

The goal of the project surrounding **ifile** is to create a Machine Learning based mail filter which is both highly accurate and efficient enough to be usable by regular e-mail users. In pursuit of this goal, we have created a system using the naive Bayes algorithm which works as a filter for the EXMH mail client. The goal of making it efficient enough for it to be usable by regular e-mail users has already been achieved. Currently, 35 **ifile** users are subscribed to a mailing list which is meant for the discussion of the ifile system. Many of these users have indicated that **ifile** is a useful tool and does, in fact, solve many of the problems inherent in non-Machine Learning based mail filters. What is left is to give some evidence that **ifile** can successfully filter mail.

3 Experimental Design

In order to evaluate the capabilities of the **ifile** system and to examine some of the issues involved in e-mail classification, we would like to perform some number of experiments based on what we believe are important issues involved with such classification. Here, we present the classifier used, along with many of the matters which are involved in e-mail classification and evaluating such a classifier.

3.1 Naive Bayes

Naive Bayes is a simple, yet effective classifier of text documents. Since our goal is not to find the most effective Machine Learning algorithm for e-mail classification, but rather to show that such algorithms can do an effective job of e-mail classification, we choose naive Bayes for its simplicity. Without further ado, we present the text classifier which is used in the **ifile** mail filtering system. A more complete description is available in chapter 6 of [Mit97].

Naive Bayes is a statistical machine learning algorithm which makes the assumption that a document is simply a set of words chosen randomly from some underlying distribution determined by the document's class. Thus, each word is considered independent of its actual location within a document. Given this model, a set of classes that generate documents, $C = \{c_1, \ldots, c_m\}$, and a document, d, which is composed of a set of words, $\{w_1, \ldots, w_n\}$, we can calculate the probability of d having been generated by class c_i as follows:

$$\mathrm{P}(c_i|d) = \frac{\mathrm{P}(c_i)\prod_{w_j \in d}\mathrm{P}(w_j|c_i)}{\mathrm{P}(d)} \tag{1}$$

where $\mathrm{P}(c_i)$ is the likelihood of a randomly chosen document belonging to class c_i, $\mathrm{P}(w_j|c_i)$ is the likelihood that a randomly generated word from class c_i is w_j and $\mathrm{P}(d)$ is the likelihood that a randomly generated document (independent of class) is d.

For a specific document, we can find the class it is most likely to be part of by computing the above equation for all classes in C and then choosing the class which has the largest $\mathrm{P}(c_i|d)$ value. Since the probability values are only used for comparison purposes, we can drop the $\mathrm{P}(d)$ term without loss of generality:

$$c_{NB} = \mathrm{argmax}_{c_i \in C}\mathrm{P}(c_i)\prod_{w_j \in d}\mathrm{P}(w_j|c_i) \tag{2}$$

To give us reasonable probability estimates in the case of sparse data, we use the m-estimate for our word probability calculations:

$$\mathrm{P}(w_j|c_i) = \frac{n_j + 1}{n + |Vocabulary|} \tag{3}$$

where n_j is the number of instances of w_j in the documents of class c_i, n is the total number of words in the documents of class c_i and $|Vocabulary|$ is the number of words we are tracking for the purposes of classification.

3.2 Electronic Mail

Untouched e-mail documents do not directly fit into the naive Bayes model. An e-mail document is not a set of words; rather, it is a stream of characters. Large portions of e-mail documents fit in to our notion of English text quite easily and the way in which such text should be translated to a set of words is obvious. However, some sections of e-mail messages are not modeled after English text and the way in which such text should be translated into a set of words is less obvious. Furthermore, some parts of an e-mail message have little or no relation to the sender and content of a message. Whether these sections should even be part of the translated set of words is questionable.

Every e-mail document is composed of at least two parts. These are the header and body. The header is a list of fields pertaining to the message. Some of these fields, such as the Subject:, From: and To: fields are pertinent to the meaning and the human understanding of the message. Other fields, such as the Received:, Date: and Message-id: fields, may have little, if anything, to do with the intended content of the e-mail message. In order that the document we consider for the naive Bayes model has an interpretation close to that perceived by the receiver, we might need to ignore some of these header fields and include others. We experiment with the effect of the removal of all but the From:, To: and Subject: headers. We will denote the removal of such headers as "header trimming."

In order to apply the naive Bayes classifier on a set of e-mail documents, we must first lex the character stream of the document into a set of words. If one only considers sequences of alphabetic characters to be words, one might undesirably separate parts of a person's e-mail address in such a way that the resulting tokens are less unique than they would have been together. If one considers sequences of space-separated characters to be words, then the underlying structure of certain character sequences (such as web and e-mail addresses) may not be revealed. In our experiments, we consider the effect of three different lexing methods on e-mail filtering. We will give the name of **alpha lexer** to the lexer which considers a word to be any sequence of alphabetic characters. **White lexer** denotes a lexer which tokenizes space, or punctuation-separated sequences of characters. Finally, the **alpha only lexer** is our name for a lexer which only tokenizes space, or punctuation-separated sequences of alphabetic characters.

Two techniques which are commonly used in tokenizing text are stemming and the use of a stoplist. The goal of a stoplist is to decrease the amount of noise in the data by eliminating words which are frequent, yet uninformative. In utilizing the stoplist, we toss any tokenized word which is found in a list of 524 common pronouns, modifiers, adverbs and verbs. Such words can commonly be found in the English language in writing relating to any subject area.

The goal of stemming is to link together words which have the same root but different endings. It is often the case that a certain word root can have many different forms associated with it. For example, a tennis-related document may contain any or all of the words `serve, service, serves, served`. We use the Porter stemming algorithm to tokenize such words as a single word. The Porter stemming algorithm converts most English words to a spelling which is common for all words with the same root. For example, the above list of words might be converted to `serv` before being accumulated and used for classification purposes.

3.3 Issues

One of the difficulties in evaluating the performance of a mail filter is in obtaining datasets which are representative of the sort of data a mail filter must classify. By its very nature, the majority of regular e-mail is private. One cannot ask individuals to give up their corpus of e-mail for inspection by the public. Another option for evaluation would be to use a set of newsgroup or mailing list documents. However, such a collection would not be representative of the document distribution normally found in a person's e-mail corpus.

In order to evaluate **ifile** in a representative setting, we asked a number of **ifile** users to execute a program which would perform a variety of experiments on their collection of stored e-mail. A total of 5 users (including the author of this paper) were able to conduct the set of experiments. The same set of experiments were also performed on a set of 20,000 newsgroup documents, 1000 newsgroup messages from each of 20 newsgroups. We call this the **20 Newsgroups** dataset for convenience. This set was included in the experiments in order to provide a basis for comparison.

The traits of the data sets on which the experiments were performed varied quite extensively. In order give a sense of the variety, we provide the number of mailboxes and e-mail messages in each user's mail corpus. In addition, we provide a metric known as *Entropy*. Entropy calculations are done over the number of documents in each mailbox and is calculated as follows:

$$Entropy = \sum_{i=1}^{m} -P(c_i) \log_2 P(c_i)$$

This gives an indication of how evenly documents are distributed amongst the user's mailboxes. If documents were evenly distributed amongst all mailboxes, $2^{Entropy}$ would be equal to the number of mailboxes.

3.4 Experiments

In total, eight experiments were planned for each user's e-mail corpus. Experiment # 1 is termed the *default* experiment and is used as a base configuration for the other experiments. For experiments other than # 1, one or more attributes of the base configuration are modified. The description of these experiments is simply an indication of the attribute which changed and a short description of how it changed. The list of experiments descriptions follows.

1. alpha lexer, stoplist used, header trimming, feature selection, no stemming
2. alpha only lexer replaces alpha lexer
3. white lexer replaces alpha lexer
4. no stoplist is used
5. stemming is used
6. no feature selection is used - all words are considered for classification purposes
7. all headers are used for classification purposes
8. white lexer replaces alpha lexer, all headers are used for classification purposes

Experiments were performed using Leave-One-Out-Cross-Validation (LOOCV) on a static set of e-mail. For each experiment, all of a user's e-mail was added to a database. Then, one-by-one, documents were removed, classified and then reinserted into the database. In this fashion, a label was generated for each document in the user's e-mail corpus. This procedure was repeated for each of the experiments. Accuracy scores were generated by dividing the number of documents labeled correctly into the total number of documents labeled.

The LOOCV method was used chosen due to efficiency concerns. Had we performed multiple runs of splitting the documents into training and testing sets, significantly more computational time would have been necessary. This may have limited the number of users who would have been able to successfully run the experiments.

4 Results

4.1 Document Classification Accuracies

Experiment results are displayed in figures 1, 2 and 3. Figure 1 lists experiment results by user and provides information relating to the nature of each user's e-mail collection. Figure 2 provides the results of the experiments in the form of a graph. This provides for easier comparison between the accuracies of various users and experiments. In figure 2, experiment results are represented as bars and are listed left to right from #1 to #8. Some users were not able to generate data for all experiments. No bar is plotted in such cases. Note that the y-axis does not run the full range of possible accuracy scores. Figure 3 displays the aggregate accuracy scores for sets of mailboxes with some minimum number of messages. The far left data point for each line indicates the labeling accuracy achieved on messages which were taken from mailboxes with 10 or more e-mail documents. Note that the x-axis is a log scale. In addition, note that User #2 did not have any folders containing at least 160 e-mail messages at the time of the experiments, so the data point for folders containing at least 160 e-mail messages was not plotted.

4.2 Observations

As can be seen in figures 1 and 2, **ifile** performs quite well on most users' e-mail corpora, achieving accuracies well above the 80% mark. The lone exception is User #5, who has a vast collection of e-mail spread across an unusually large number of mailboxes. This out-lier may be related to the apparent correlation between Entropy and classification accuracies amongst the private e-mail data sets. This correlation may be due to the inherent difficulty in classifying documents across a large set of classes. In order to achieve better performance on such data sets, it may be necessary for the classifier to develop some sort of mailbox hierarchy.

When comparing individual experiments, one can see that accuracy scores for experiment #2 are never above the scores for experiment #3. The only difference between the two experiments is the choice of lexer. Experiment #2 uses the **alpha only lexer**, while experiment #3 uses the **white lexer**. The poor performance of the configuration used for experiment #2 may be caused by the fact that the **alpha only lexer** does not lex e-mail addresses into tokens. As it is fairly common for people to organize mail according to sender of e-mail messages, the **alpha only lexer** discards valuable classification information, so it is little surprise that it performs poorly compared to other lexers.

One should also note the poor accuracy scores for experiment #6. All users which performed experiment #6 found classification accuracies which were consistently below the accuracies achieved in experiments #1, #5 and #7. Experiment #6 retains all header fields for the classification of e-mail messages while experiments #1, #5 and #7 only retain the "From:", "Subject:" and "To:" header fields. As many of the other e-mail headers do not pertain to the sender or content of an e-mail message, it is likely that they only add noise and detract from the purpose classification. It should be noted that the reason the **20 Newsgroup** data set performs well when all of the headers are included is due to the fact that the class label for each document is listed in the "Newsgroup:" header.

User #1

Ex #	Accuracy
1	88.83%
2	90.12%
3	90.93%
4	84.97%
5	88.91%
6	85.84%
7	89.95%
8	92.16%

$2^{Entropy} = 15.6$
27 mailboxes
2715 e-mail messages

User #2

Ex #	Accuracy
1	89.81%
2	87.66%
3	87.66%
4	89.27%
5	89.27%
6	88.47%
7	90.34%
8	

$2^{Entropy} = 8.5$
16 mailboxes
373 e-mail messages

User #3

Ex #	Accuracy
1	90.68%
2	88.24%
3	88.85%
4	89.61%
5	91.45%
6	89.46%
7	90.07%
8	

$2^{Entropy} = 6.0$
13 mailboxes
655 e-mail messages

User #4

Ex #	Accuracy
1	86.48%
2	81.94%
3	83.44%
4	83.87%
5	86.14%
6	79.46%
7	81.89%
8	79.96%

$2^{Entropy} = 14.2$
33 mailboxes
4447 e-mail messages

User #5

Ex #	Accuracy
1	67.16%
2	66.53%
3	69.69%
4	59.17%
5	68.10%
6	
7	70.52%
8	71.28%

$2^{Entropy} = 60.8$
132 mailboxes
5665 e-mail messages

20 Newsgroups

Ex #	Accuracy
1	83.19%
2	83.45%
3	83.61%
4	82.42%
5	82.85%
6	86.10%
7	86.44%
8	89.24%

$2^{Entropy} = 20$
20 mailboxes
19997 e-mail messages

Figure 1: Full listing of experiment accuracies and data set information.

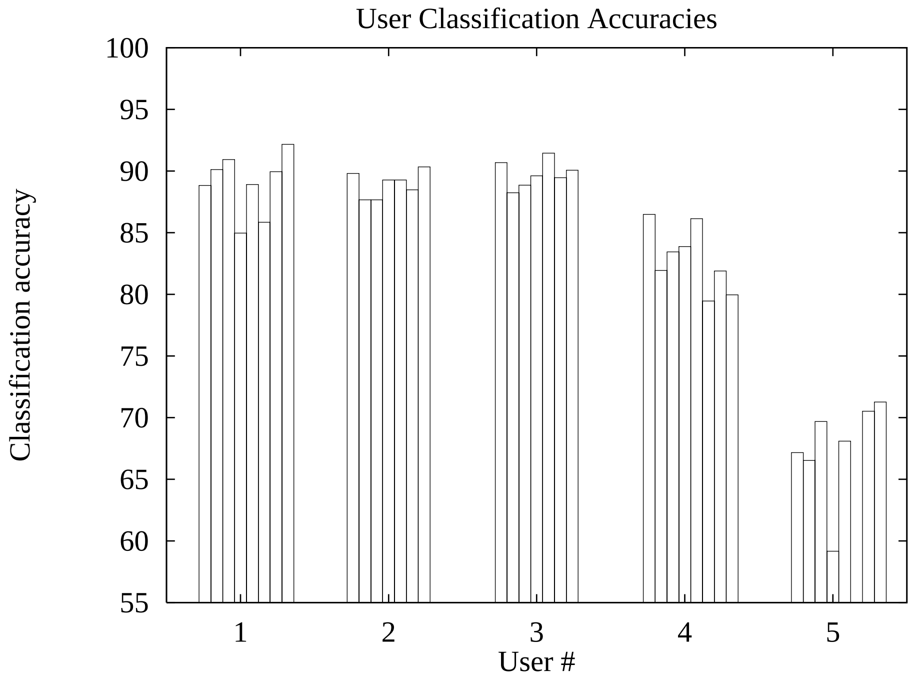

Figure 2: Experiment classification accuracies grouped by user. Experiment #1 is at the left end of each grouping. Experiment numbers increase to the right.

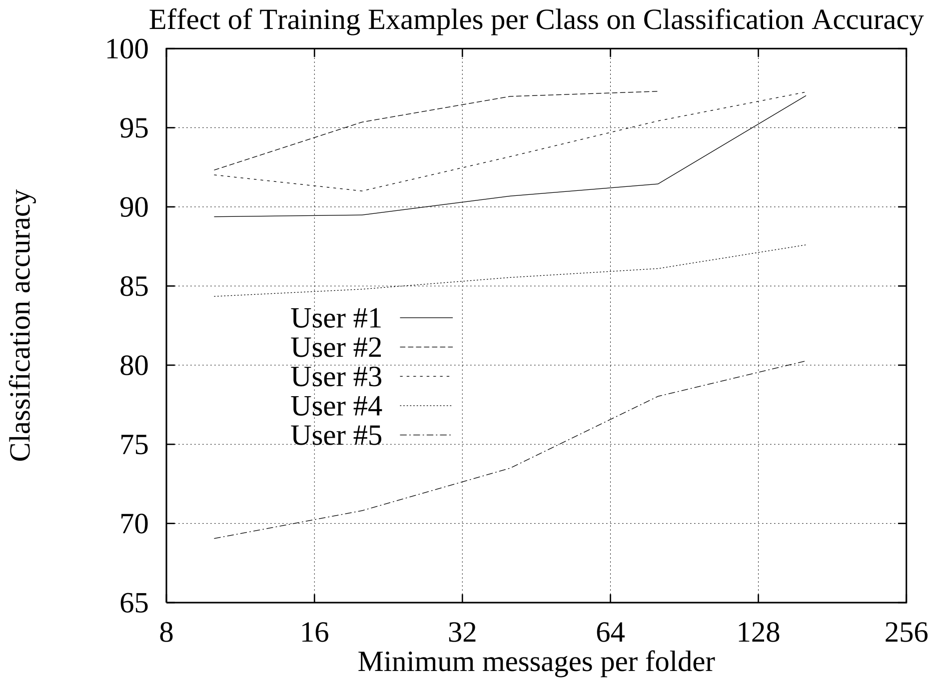

Figure 3: Aggregate classification accuracies when only considering classification scores for mailboxes with a minimum number of messages

5 Discussion

User feedback and the set of experimental evidence provided in this paper give strong evidence to back the proposition that mail filtering has something to gain from text-based Machine Learning techniques. **ifile** is capable of achieving high classification accuracies while eliminating many of the annoyances of traditional mail filters. No longer does the user need to spend time creating and updating rules and she is no longer restricted to organizing mail according to keyword matching.

However, even with the apparent success, there is still room for improvement. **ifile** appears to have not performed well in the case of a user with a large collection of e-mail spread across a vast array of mailboxes. And even though **ifile** performed well in most other cases, it would be useful to increase accuracy scores beyond their current marks.

In order to achieve better filtering for users which maintain a large set of mailboxes, it might be necessary to construct mail clients which more easily facilitate the organizing of mailboxes into complex hierarchical structures. The mail client for which **ifile** was created, EXMH, makes it somewhat difficult to maintain a deep mailbox hierarchy. This problem might also be solved by applying a different Machine Learning algorithm; one which is better at handling such large numbers of target classes.

Motivated by the quest to achieve greater filtering accuracy, one interesting area of possible exploration would be the combination of a rule-learning algorithm with a text classification algorithm such as naive Bayes. Rule-based learning algorithms, such as CN2, used in [Pay94], often allow fine control of the tradeoff between precision and recall. If all messages to be filtered were first passed through a high-precision rule-based classifier, we might be able to give highly accurate labels to a portion of the documents before handing them on to a less precise naive Bayes classifier.

It might also be useful to detect threads as in [LK97] under the assumption that all messages which are part of a certain thread belong to the same mailbox. If a layer is added to the filter which can determine whether an e-mail message has a parent, and if so, which mailbox that message is located in, then one could accurately label additional documents before passing them to naive Bayes for classification.

Another area of possible future research is in tuning the classifier so that it is biased toward the content in e-mail documents which were received during a certain time period. One could easily imagine that the validity of basing filtering on any particular e-mail message is directly proportional to the age of the document. An example of this would be a discussion group where the discussion focus shifts between different topics. As this decay of a document could easily depend on the mailbox in which it belongs, one might want to learn decay rates for each of the mailboxes for a particular user. This would allow the classifier the ability to make better use of the training examples it has available.

The one problem with traditional mail filters which is not addressed in this paper is that of mailbox organizational changes. When a user decides to rename mailboxes, or reorganize her mailbox hierarchy, she must rewrite her filtering rules to incorporate the changes. In the case of **ifile**, the mail filter takes no special action to recognize this severe organizational change. Unless the user wishes to deal with a classifier which is not very well suited for her current mailbox organization, she must initiate a rebuilding of the **ifile** database according to the currently available e-mail corpus. It would be quite useful if the filter were able to detect such a change and rebuild the database accordingly.

6 Acknowledgements

We thank Sean Slattery, Kamal Nigam, Victor Huang and Dave Robinson for allowing experiments to be run on their e-mail collections. We thank Andrew McCallum, Diego Zamboni, Harry G. McGavran Jr., Carl Staelin, Valdis Kletnieks, Dave Robinson and Chris Browne for **ifile** code contributions and bug reports. We thank the members of the Carnegie Mellon University Text Learning group for helpful advice and constructive criticism. Most of all, we thank Tom Mitchell for his kind guidance, never ending patience and love for learning and exploration.

References

[Coh96] William Cohen. Learning rules that classify e-mail. In *AAAI Spring Symposium on Machine Learning in Information Access*, 1996.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, 1998.

[Lan95] Ken Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.

[LK97] David D. Lewis and Kimberly A. Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

[Mit97] Tom Mitchell. *Machine Learning*. McGraw-Hill Companies, Inc., 1997.

[Pay94] Terry Payne. Learning email filtering rules with magi, a mail agent interface. 1994. MSc Thesis, Department of Computing Science, University of Aberdeen, Scotland.

[YP97] Y. Yang and J.P. Pedersen. Feature selection in statistical learning of text categorization. In *Fourteenth International Conference on Machine Learning*, 1997.