Email Folder Classification using Threads

Bryan Klimt and Yiming Yang
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

While automatic classification of email is obviously a useful task to study, it is not obvious how to best utilize the rich metadata specific to email to improve the quality of the classification. In this paper, we propose a simple algorithm for using email *threads* to improve the precision of a personal email assistant's automatic folder classification. We evaluate the approach on a large email dataset, and confirm the intuition that email threads can be used for high precision but low recall. Futhermore, we experimentally verify the intuition that using threads is hampered in its improvement of a classifier by the redundancy of their information.

1 Introduction

The volume of email people receive today has increased to levels such that the development of automatic email organization tools will soon be critical for many users to accomplish their daily tasks. These tools include spam filters, email prioritizers, and plugins for automatically sorting mail into user-defined folders or labels. One of the greatest challenges in creating these tools is determining how to represent emails to obtain the best performance from machine learning algorithms. While document representations for standard text categorization have been extensively studied and tested, the rich metadata relating to email allows for many possibilities that have not yet been explored. We will give a brief overview of the feature representations that have been proposed, followed by an in-depth examination of the usefulness of in folder classification of a particularly interesting feature unique to email, *threads*.

(Manco et al., 2002) identified three classes of features present in email: unstructured text, semi-structured text, and numeric data. Others have suggested a fourth type of feature, relationships, such as those between emails and between users. Unstructured text in email consists of fields like the subject and body, which may contain any natural language text. These fields can easily be treated in email the same way they are in other text categorization domains, such as standard bag-of-words representations (Cohen, 1996; Manco et al., 2002; Rennie, 2000). Predictably, many have found that some of these natural language text fields are more important than others in classifying email (Diao et al., 2000; Manco et al., 2002). For instance, (Klimt & Yang, 2004) found that the body of an email is the most useful text field feature for folder classification, yielding much better performance than the subject field.

Semi-structured text fields (called categorical text by (Manco et al., 2002)), such as "to" and "from", are different that unstructured text fields because the text they contain is very well defined, both syntactically and semantically. So far, these fields have been treated like unstructured text fields, as other methods for using them are non-obvious. (Klimt & Yang, 2004) found that, even used in this simple way, categorical text fields are useful in email folder classification, especially when combined with unstructured data, such as the body.

Numeric Data in email includes such features as the message size, number of recipients (Manco et al., 2002), and counts of particular characters (Diao et al., 2000). While these features may be useful for spam identification, they have so far been shown to provide little benefit for folder classification.

The fourth class of email features, relationship information, has not yet been as extensively studied as the other types of email features for its use in email classification. Relationships, such as those between an email message and other objects, such as users, folders, or other emails, could play an important role in the future of email management tools, much as the use of hy-

perlinks has improved web classification and retrieval. In this paper, we examine one of these relationships, that of *thread membership*.

A thread is considered to be the chain that relates a set of emails sent among a set of users discussing a particular topic. Although threads are implicit, rather than explicit, they provide more rich context for each email than what is explicitly mentioned by the email in isolation. Several methods for deducing this information have been proposed. (Lewis & Knowles, 1997) used message reply trees as an approximation of thread structure, such that two messages are considered to be in the same thread if one is a reply to the other. The relationship is then propogated transitively. (Klimt & Yang, 2004) also used this definition, and proposed another algorithm for identifying email threads based on it. (Murakoshi et al., 1999), on the other hand, used linguistic analysis to identify email threads. Unfortunately, that is a difficult natural language problem, and is difficult to evaluate.

In this paper, we indentify threads using the algorithm given in (Klimt & Yang, 2004). We then attempt to use those threads to improve the quality of folder classification, as described in the next sections.

2 Methods

To recover the thread structure in the email, we define threads such that two emails are in the same thread iff:

1. The subjects of the emails are the same, after stripping off "RE:" and "FWD:" from the beginning and converting to lower-case, and

2. The "to" and "from" fields of the emails have at least one user in common. (Their intersection is non-empty.) Note that in the case of a single user, this is trivial, since that user will have sent or received every email in his folders.

The set of messages with empty subjects were not considered to be a thread, even though they otherwise fit this definition. We made no attempt to test the quality of this thread definition, other than the indirect evaluation of measuring its effectiveness for classification. The main reason for not evaluation the threads is that threads are rather subjective, and user judgments may disagree. Lewis (Lewis & Knowles, 1997) had used the "in-reply-to" headers in email messages as the ground truth about thread membership to test his thread detection algorithm. Unfortunately, it appears that some current email clients do not use this header, as the Enron corpus has very few messages with it. Also, our definition could not be evaluated against the results from the previous paper, as the corpus used in that paper is no longer available.

To determine the effectiveness of thread information for email folder classification, we compared several email classification algorithms, comparing those that use threads with those that do not. These algorithms are labled as FROM, TO, SUBJECT, BODY, ALL, COMBO, THREAD, COMBO+THREAD, and ALL+THREAD. The details of each are outlined in the next sections.

FROM, TO, SUBJECT, BODY, ALL

The FROM, TO, SUBJECT, and BODY classifiers are based on standard text classification techniques (**support vector machines**) using the given sections of the email, treated as bags-of-words. The ALL classifier is similar, but uses the text from all four of the other fields, dumped into one bag-of-words.

We are using support vector machines for these classifiers, based on their previous success in email folder classification. (Brutlag & Meek, 2000) found that SVM outperformed other learning algorithms, such as Naïve Bayes, for email folder classification, especially for folders that contain a large number of messages. (Klimt & Yang, 2004) found that there is a correlation between the volume of email a person receives and how much email they have in each of their folders. This evidence suggests that SVM should perform well for users with a large amount of email, for whom automatic email management tools will be most useful.

COMBO

The COMBO classifier is based on the algorithm given in (Klimt & Yang, 2004), which combines the scores of the FROM, TO, SUBJECT, and BODY classifiers based on their performance. When a new email is given for training, first each of the component classifiers is trained using the message. Then, each classifier is asked to use its learned model to make a prediction for every one of the messages that have been used for training so far. **Linear regression** is then used to learn weights for each of the classifiers in order to minimize the prediction error for the training set. Thus, good classifiers are given large weights, while poor ones are given relatively smaller weights. During the testing phase, the COMBO classifier simply combines the predictions of its component classifiers using these weights.

THREAD

The first new algorithm being proposed by this paper is the THREAD classifier, which makes predictions about email folders using only email threads information. This classifier is based on a very simple and intuitive probabilistic model. Since thread membership is defined to be transitive, threads are mutually exclusive. That is, a message belongs to exactly one thread. Otherwise, the threads would have to be unified, and be considered a single thread. In other words, for any two threads, $p(thread1 \wedge thread2|email) = 0$. Given this assumption of mutual exclusivity, we can express the folder classification task as follows:

$$p(folder|email) = \sum_{t \in threads} p(folder|t) \cdot p(t|email)$$

The probability of an email being in a particular folder is the probability of that email being in a thread times the probability of an email in that thread being in that folder. In order for this notation to be useful, we must define each of its subcomponents. For the $p(t|email)$, we simply use the definition given in (Klimt & Yang, 2004). That is, we define $p(t|email) = 1$ if the email is in the thread, based on the thread definition given above, and $p(t|email) = 0$ otherwise. For the other component, $p(folder|t)$, we use the maximum likelihood estimator:

$$p(folder|t) = \frac{count(email \in t \wedge email \in folder)}{count(email \in t)}$$

Simply put, the probability of an email being in a folder, given that the email is in thread t, is the percentage of emails in t that are in the folder. Notice that there is a degenerate case here. When the email does not belong in any of the existing threads, no prediction can be made. This means that the THREAD classifier is prone to low recall, as it may not be able to made a classification for many messages.

COMBO+THREAD, ALL+THREAD

The COMBO+THREAD and ALL+THREAD classifiers are similar algorithms based on augmenting existing classifiers using the THREAD classifier. Since the THREAD classifier would intuitively have high precision and low recall, we simply trust the THREAD classifier whenever it makes a prediction, and use the other classifier whenever it cannot. An example of one of these algorithms is given in pseudocode in Listing 1.

Listing 1 COMBO+THREAD Classifier Algorithm

```
thread_prediction =
  thread_classifier.classify( email );

combo_prediction =
  combo_classifier.classify( email );

if score of thread_prediction >= .5
  return thread_prediction;
else
  return combo_prediction;
```

3 Dataset

When evaluating algorithms for automatic email folder classification, it is desirable to choose a test dataset that accurately models the users of the proposed system. For that reason, we must find data that is both realistic and has many users with a large amount of email. Users with a lot of email would be the ones most benefitted by automatic classification tools. Furthermore, the users in the dataset must have manually sorted their email into folders or manually applied classification labels (using their own organizational strategies). To evaluate our performance, we can then compare our classifications with their manual labels.

To meet these goals, we have chosen to use the freely available Enron Corpus (Klimt & Yang, 2004), as it is the only email corpus that we are familiar with which meets these requirements. The Enron corpus, which became available as part of court proceedings against the Enron corporation, contains 200,399 messages of 158 users. Almost all of these users use folders for sorting their email, and they exhibit a wide variety of organizational strategies. This dataset is currently available online at `http://www-2.cmu.edu/~enron/`.

Unfortunately, time constraints and the long, unpredictable, runtime of SVM, we limited us in the amount of data we could use for our experiments. We chose to use the 100 of the 158 users in the Enron corpus with the least email. Although we would like our algorithms to be useful for users with the most messages, the techniques described in this paper were found to be intractable for some of the users in this dataset with hundreds of thousands of messages. Improving the efficiency of these techniques will therefore be an important step in future work. Furthermore, the emails in the Enron dataset range from 1997 to 2002, so in practical applications, the algorithm may have a considerable amount of time to run for users with a lot of email. The largest user in our dataset still has about 2000 messages.

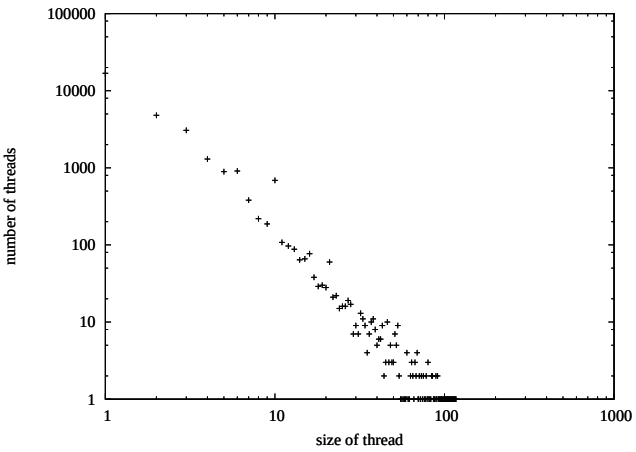

Figure 1: Thread Size Distribution

(Klimt & Yang, 2004) gave an initial analysis of the thread data present in the Enron dataset, which we will summarize here.

Out of the total 200,399 messages in the Enron corpus, our method found 30,091 threads in the corpus which could be used for classification. This accounted for 123,501 (62%) of the 200,399 messages. The average average thread size 4.10 messages, but unfortunately, the median thread size is only 2. So, there are a few large threads in the corpus, and many small threads. The distribution of thread sizes is given in table 1.

Theoretically, larger threads should be more useful for classification, since they provide more information about the relationships of a message. Unfortunately, larger threads are much less common. More important than the size of the thread, though, is the information the thread can provide. The average number of folders containing the messages of a thread is 1.37. This means, given an average thread, the messages in that thread are distributed among only 1.37 folders. This suggests that our MLE-based classifier should be very precise.

The major drawback of this thread information is that it may be redundant when used with the other kinds of evidence discussed in this paper. Since subject words are used to detect threads, a thread based classifier may not provide any information not already available, if it is used as just another feature. One example of a redundancy problem is with the largest thread in the Enron corpus, messages with the subject "Demand Ken Lay Donate Proceeds from Enron Stock Sales" belonging to user "lay-k". There are 1124 messages in this thread and they are all in the same folder (Deleted Items)! This would be incredible evidence for classification by itself. However, all of the messages in the thread are virtually identical; they appear to be SPAM. Since the messages are identical, there is already incredibly strong evidence from other features, without even detecting the thread.

4 Experiments

We designed our experiments in an effort to best reflect the circumstances in which an automatic email classifier would be useful for a real user. In this scenario, each time an email arrives at the user's computer, the tool chooses the folder (or label) that it thinks that email belongs to. The user then has an opportunity to either confirm or reject the suggestion. Based on the user's feedback, the model is retrained. Besides being more realistic than the normal splitting of training and testing data, this experimental setup is necessary to show the benefits of using email threads for classification. If an email dataset were split chronologically into training and testing sets, the only useful threads would be those that contained messages in both sets. However, most threads are short enough in duration that they do not span both datasets, so that sort of setup would not be able to take advantage of them.

The basic design of the classifiers is given in section 2. For the SVM classifiers, we used a one-vs-rest approach for multiclass classification. The exact settings and parameters were basically the same as those used in (Klimt & Yang, 2004). However, the algorithms had to be modified to be trained incrementally, reflecting the new experimental setup as described above. Word canonicalization was performed using the Porter stemmer for English (Porter, 1980). The feature vectors were then weighted using the "ltc" term-weighting scheme as defined by (Buckley et al., 1995), such that the value w_{ik} for term k in the feature vector for email i is given by

$$w_{ik} = \frac{(\log(f_{ik}) + 1.0) * \log(N/n_k)}{\sqrt{\sum_{j=1}^{t}[(\log(f_{ij}) + 1.0) * \log(N/n_j)]^2}}$$

where f_{ik} is the frequency of term k in email i, N is the number of emails that have been seen, and n_k is the total number of emails we have seen so far which contain the term k. Note that this term-weighting scheme results in feature vectors that are normalized to unit length.

5 Results and Analysis

Several measures of our experimental results are given in table 1, and shown graphically in figures 2 and 3. Accuracy is defined as the percentage of messages the classifier classified correctly. Recall and precision are

Classifier:	From	To	Subject	Body	Thread	All	All+ Thread	Combo	Combo+ Thread
Accuracy	.65	.62	.62	.64	.24	.66	.67	.67	**.67**
Micro Recall	.65	.62	.62	.64	.24	.66	.67	.67	**.67**
Micro Precision	.71	.70	.72	.76	**.81**	.78	.77	.78	.77
Macro Recall	.44	.39	.40	.40	.16	.41	.43	.45	**.45**
Macro Precision	.66	.69	.69	.77	**.82**	.78	.75	.72	.69
Micro F1	.68	.65	.66	.68	.35	.70	.71	**.71**	.71
Macro F1	.50	.47	.47	.48	.26	.50	.51	**.53**	.53

Table 1: Classification Results

defined as in (Yang & Pedersen, 1997):

$$recall(f) = \frac{count(e \in f \wedge prediction(e) = f)}{count(e \in f)}$$

$$precision(f) = \frac{count(e \in f \wedge prediction(e) = f)}{count(prediction(e) = f)}$$

for every email e and folder f. F1 score is the harmonic mean of recall and precision:

$$\frac{1}{2} \cdot \left(\frac{1}{recall} + \frac{1}{precision} \right)$$

For micro-average scores, the score for each email was averaged. For macro-average scores, the score for each folder was calculated, and the average score for all of the folders was calculated.

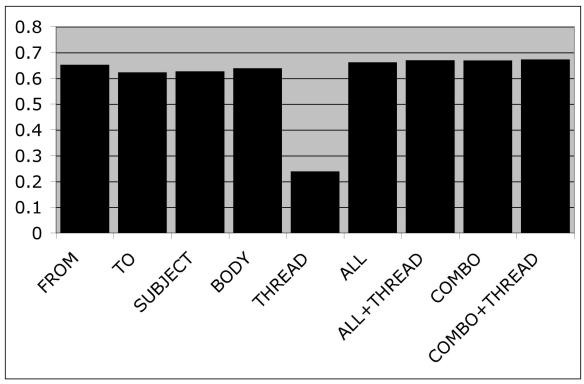

Figure 2: Accuracy of Classification using Different Types of Information

While accuracy can sometimes be a misleading measure of classification performance, it is given here because of its easy interpretability. That is, we can see that an automatic email classifier based on these techniques will be correct between 60% and 70% of the time. The THREAD classifier obviously has much lower performance by itself than the other techniques, but as noted above, this is because it does not make any predictions for the large number of emails that are not in threads.

One interesting observation is that the difference in quality between the ALL and COMBO classifiers is much smaller here than in (Klimt & Yang, 2004). We believe this is because the COMBO classifier has difficulty learning weights as the relative quality of the classifiers changes much during the incremental training.

As for the hybrid classifiers, COMBO+THREAD and ALL+THREAD, we see that they do not have significantly higher accuracy than the methods not using threads. To see why, we must examine the recall and precision in figure 3. As expected, the precision of the thread-based classifier is quite high, significantly higher than all of the other classifiers. However, the recall is (also unsurprisingly) quite low. So, given the high precision of the thread classifier, why does it not provide increased precision to the hybrid classifiers? In fact, it actually *hurts* their precision. The reason for this is as predicted in (Klimt & Yang, 2004). While the thread-based classifier is very precise, its information is highly redundant with respect to the other classifiers, such as the one based on the *subject* field.

To verify this, we analyzed the decisions made by the COMBO and THREAD classifiers, to see if THREAD decisions really were redundant. It turns out that, for this dataset, the COMBO and THREAD classifiers made the same prediction 93% of the time, meaning they are highly redundant. When they agreed, their accuracy was 89%, which is higher than any other classifier. Of the instances where they disagreed, the THREAD was actually right less often, only 31% of the time, while the COMBO classifier was correct for 60%. This means that, although the thread classifier is the most precise classifier, it is not as precise as COMBO on the emails where they disagreed. More work needs to be done in determining automatically when THREAD is right in the case of disagreements, but even then the redundancy means it can only be effective for a small fraction of the total emails (31% of 7% = 2%).

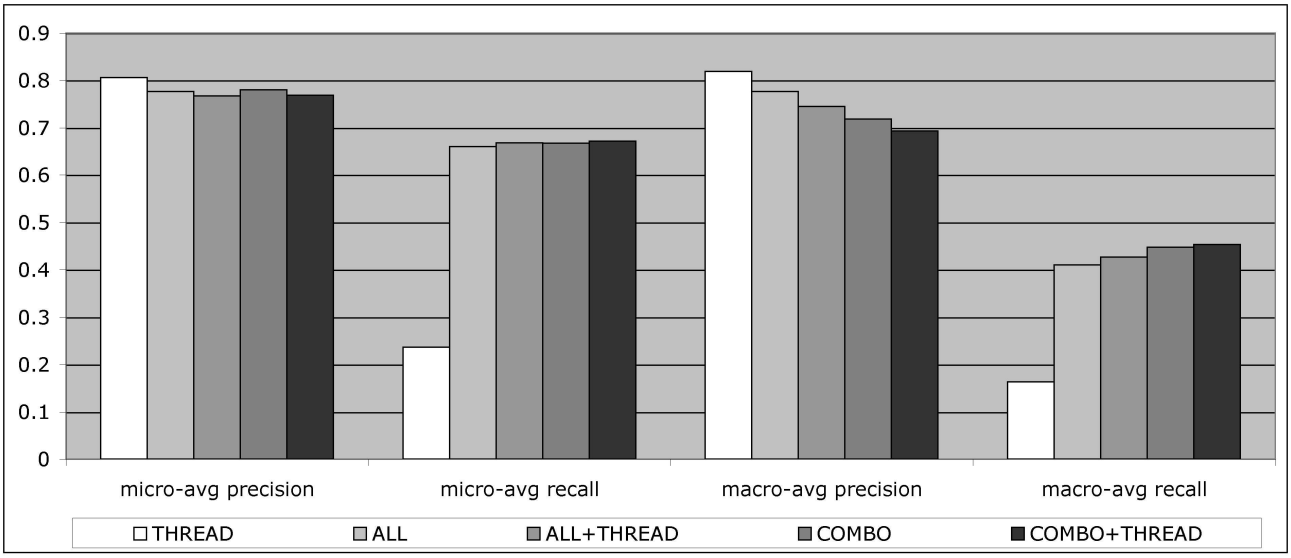

Figure 3: Recall and Precision of Different Approaches

6 Conclusions and Future Work

In this paper, we have given a simple and intuitive method for using email threads for the task of email folder classification. While we have shown that methods based on threads can yield very high precision, it is also clear that their use is limited, given their redundancy when used in concert with other, text-based classifiers.

The biggest challenge in our future work is actually improving the efficiency of our SVM-based classifiers to meet the needs of users with a significantly larger amount of email. There are several proposed techniques for doing this, which we may address in a future work. Also, it may be possible that some other techniques which use email threads (or the conversation graph structure) may better improve email folder classification. This will likely require a more creative approach, possibly using other kinds of relationship data, such as the connections between users.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHC030029.

References

Brutlag, J., & Meek, C. (2000). Challenges of the email domain for text classification. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 103–110).

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). Automatic query expansion using SMART: TREC 3. *Proceedings of the Third Text REtrieval Conference* (pp. 69–80). NIST Special Publication.

Cohen, W. (1996). Learning rules that classify e-mail. *Proceedings of the 1996 AAAI Spring Symposium in Information Access* (pp. 124–143).

Diao, Y., Lu, H., & Wu, D. (2000). A comparative study of classification-based personal email filtering. *Proceedings of the Fourth Pacific-Asia Conference Knowledge Discovery and Data Mining* (pp. 408–419).

Klimt, B., & Yang, Y. (2004). The Enron corpus: A new dataset for email classification research. *Proceedings of the Fifteenth European Conference on Machine Learning* (pp. 217–225).

Lewis, D., & Knowles, K. (1997). Threading electronic mail: A preliminary study. *Information Processing and Management*, *33*, 209–217.

Manco, G., Masciari, E., Ruffolo, M., & Tagarelli, A. (2002). Towards an adaptive mail classifier. *Proceedings of Tecniche di Intelligenza Artificiale per la ricerca di informazione sul Web (AIIA).*

Murakoshi, H., Shimazu, A., & Ochimizu, K. (1999). Construction of deliberation structure in email communication. *Proceedings of the Pacific Association for Computational Linguistics* (pp. 16–28).

Porter, M. (1980). An algorithm for suffix stripping. *Program*, *14*, 130–137.

Rennie, J. (2000). ifile: An application of machine learning to e-mail filtering. *Proceedings of the KDD00 Workshop on Text Mining*.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. *Proceedings of ICML-97, 14th International Conference on Machine Learning* (pp. 412–420).