Automatic Induction of Rules for e-mail Classification

Elisabeth Crawford
Basser Department of Computer Science
The University of Sydney
NSW 2006 Australia
ecrawfor@cs.usyd.edu.au

Judy Kay
Basser Department of Computer Science
The University of Sydney
NSW 2006 Australia
judy@cs.usyd.edu.au

Eric McCreath
Department of Computer Science
The Australian National University
ACT 0200 Australia
ericm@cs.anu.edu.au

Abstract

Many users receive large amounts of email. Since a substantial part of that mail is kept for future reference, it is unsurprising that many mail tools allow users to create filtering rules that automatically do actions like saving mail in suitable folders. Unfortunately, most users do not make much use of this facility.

This paper describes the i-ems project which explores approaches to building a system to assist users in managing electronic mail. It does this by learning rules for classifying email so that it can assist the user in filing messages. This classification is also a precondition to automated action on behalf of the user and we are particularly interested in using it as part of a smart personal assistant.

We begin with a review of similar work and identify the core issues for the i-ems project. We relate the past work to classic information retrieval work in text classification and discuss the major new issues that we need to address. We present a simple user interface which automatically learns rules to improve the user's overall interaction with the email manager. We also present empirical results which compare four different learning approaches as email is progressively provided.

Keywords Machine Learning, User Modelling, Text Classification, Inductive Logic Programming, email.

1 Introduction

The number of people using email has grown dramatically over the last ten years and users receive increasing volumes of email. This is partly due to the growth in email use as a means of internal communication within organisations. It is also due to the growth in spam or junk mail. There are several problems associated with growing volumes of email. An important one relates to the increasing amounts of time that people need for reading and processing email.

This paper is concerned with the problems associated with the management of email messages that the user wants to keep and the filtering of junk email. Many email messages are simply read, then deleted. Many others are answered, then deleted. This leaves many mail items which are kept. One strategy for managing this email is to archive each piece into an appropriate folder. Choice of folder may depend on many factors including aspects such as the sender and nature of the email. For example, email from your manager may be filed into your *"manager"* folder. This task is non-trivial. For example, one study by Whittaker and Sidner [20] of 20 workers found the average inbox had 2482 items. The average number of filed items was only 858. Within the study, some users left almost all their email in the inbox.

The proliferation of junk email since the mid-1990's means it creates an annoying and time wasting task for email users as they filter junk email, commonly referred to as "spam". Cranor and LaMacchia [7] report that the 'spam rate' on most days in the Spring and early Summer of 1997 for the AOL system was 30%. Their own study of a corporate network found that 10% of incoming email was spam. Apart from the cost in terms of time for the user to determine the spam from the legitimate email, spam can also be harmful and offensive. For example in an analysis of 400 of the spam emails the corporate network received, Cranor and LaMacchia found that 11% were 'adult' or sexually oriented.

Proceedings of the Sixth Australasian Document Computing Symposium, Coffs Harbour, Australia, December 7, 2001.

Many email programs[1] support filtering rules which can automate various mail management tasks: automated filing, deletion, replies. These rules can be expressed in terms of strings appearing in different parts of an email message. To handle an email item, the rules are evaluated in order and the first rule that applied to the item triggers the email client to move the message into the associated folder. The difficulty with rules is that the process of composing a rule is cognitively demanding and there is a real, potentially unacceptable risk of misfiling mail. Generally, users seem to avoid customising software [11]. Email classifiers aim to take the burden of classification, be it in the form of manual filing or rule writing from the user.

The aim of i-ems (Intelligent Electronic Mail Sorter) is to investigate the automatic induction of email filtering rules. This might operate as an advising agent which suggests likely folders for storing a message including a "Junk" folder for spam. Alternatively, it might handle mail automatically, either at the stage of mail arriving or after the user has displayed it. At the same time, we want to provide an intuitive interface which enables users to scrutinise both the rules constructed for them and the *reasoning underlining the system's construction of these rules.*

2 Previous Work

A variety of approaches have been taken to address the problem of automating email classification. Most of these approaches address, either the general classification of email or the specific filtering of junk email. Although this can be viewed as a special case of text categorisation, this section reviews only work in management of email. As we will discuss in the next section, this domain has special characteristics that distinguish it from the classic work in text categorisation, even the seemingly similar work on the Reuters-21578 dataset.

2.1 General classification

A number of systems have examined different ways of classifying email using Machine Learning and IR approaches. Some of these systems are described below in approximate chronological order.

- ***Learning Rules that Classify E-Mail*, by Cohen** [**6**]. Cohen uses the RIPPER learning algorithm that induces rules that spot keywords for classifying email. The paper compares this "keyword spotting" approach with an IR method, based on TF-IDF weighting. Both approaches show similar accuracy, RIPPER 87%- 94% and TF-IDF 85%-94%. However, Cohen argues that keyword spotting is more useful as it induces an understandable description of the email filter.

- ***MailCat*, by Segal and Kephart** [**19**]. This system uses a TF-IDF approach which computes weighted vectors for each folder based on word frequencies, then a distance measure is used to estimate the similarity a new message has with each folder. When new messages were directly filtered into the most similar folder an error of 20% to 40% resulted. A later version took a slightly different approach: *MailCat* would give the user an option of the three most likely folders. This significantly improved performance. The user could then archive the email with a single mouse click on one of the three folder buttons.

- ***iFile*, by Rennie** [**17**]. Rennie uses a naive Bayes approach for text classification. The *iFile* works as a filter for the EXMH mail client. The system applies stemming and makes use of a stop-list. A number of *iFile* users were provided a program that performs a series of experiments on their own email folders; this helped address privacy issues relating to email. However, it also limits the ability of other researchers to perform a comparison with other approaches. On these folders Rennie achieved 89% accuracy using a naive Bayes classifier.

- ***SVM*, by Brutlag and Meek** [**4**]. Brutlag and Meek compare the performance of Linear Support Vector Machines (SVM), TF-IDF and a Unigram Language Model for email classification. They preprocessed the data using a Zipf filter which removes very rare words as well as very common words. They found that classification accuracy varied more between mail stores than between classifiers and that no one classifier was consistently superior. Brutlag and Meek also looked at the effect of folder size on accuracy, finding that TF-IDF offered the best performance for sparse folders while SVM was very accurate on dense folders. They reported that the accuracy of Linear SVM ranged between 70% and 90%; the Unigram Language Model between 65% and 90% and the TF-IDF approached between 67% and 95%, depending on the store of email used.

- ***Magi* by Payne and Edwards** [**13**]. Payne and Edwards have implemented Magi (Mail Agent Interface) on top of a Unix mail System. Using Magi they have examined two different techniques for sorting email, CN2 a rule induction algorithm and IBPL1 a modified version of the K-nearest Neighbour algorithm

[1]Including the widely used programs Netscape and Microsoft Explorer.

which uses Memory Based Reasoning. These approaches are used to predict actions e.g forward, delete and file in folder X. Depending on the confidence Magi has in its prediction it will either carry the action out automatically, suggest the action to the user and see if they agree, or make no suggestion at all. Payne and Edwards found that 26% of the time when using CN2 no suggestion was made and 22% of the time when using IBPL1. When it did make suggestions, for CN2 65% were correct and for IBPL1 57%.

- ***Re:Agent*, by Boone** [3]. Unlike other approaches *Re:Agent* is divided into two distinct stages. In the first stage, features are either learnt from collections of email messages[2] using a TF-IDF approach or they are constructed by users providing keywords. This permits the user to guide the agent without explicitly formulating rules. The second stage uses the features, constructed by the first stage, for learning actions. Boone investigates both neural networks and nearest neighbour approaches for this learning. Both these learning approaches are aided by the significant reduction in the dimensionality gained by first constructing the features. However unlike other approaches described Re:Agent only classifies email into two categories, 'work' and 'other', so its results cannot easily be compared with the work of other authors. However Boone found that using features achieved 98% accuracy, while the standard IR approach had 91% accuracy.

2.2 Junk Mail

There have been attempts in a number of different areas to reduce the problem of spam. These range from suggestions of legal and regulatory measures[7], to content based filtering and multiple email addresses or alias' schemes [8, 10].

There has been some good progress in the use of Machine Learning and Information Retrieval approaches in detecting spam email, based on the content of the email header and body. Some of these approaches are described below.

- ***SpamCop*, by Pantel, and Lin** [12]. The objective of *SpamCop* is filtering junk email, hence its name. The system implements a Naive Bayes approach making use of both stemming and a stop-list. The stop-list is dynamically created by words that either occur less than 4 times in all the messages or by words that occur a similar proportion of time in both junk and legitimate messages. Note that this is different from classical Information retrieval stop lists which contain very common words. Pantel and Lin compared their system with Cohen's keyword-spotting approach *Ripper*. They found that Naive Bayes out performed RIPPER with an an accuracy level of 94% compared to 86%.

- ***Bayesian Approach*, by Sahami, Dumais, Heckerman, and Horvitz** [18]. This system learns to filter junk email using a Bayesian Approach. As it is less desirable that legitimate messages are labelled as junk, than junk email is labelled as legitimate, a 99.9% threshold is used for classifying a message as junk. The investigation found that the incorporation of both phrases and domain specific features enhanced the system's performance. These phrases and domain specific features were constructed by hand. Without the handcrafted features their Bayesian approach achieves recall of 94.3% on junk and 93.4% on legitimate mail and precision of 97.1% on junk and 87.7% on legitmate mail.

- ***Genetic Approach*, by Katirai** [9]. Katirai compares a Genetic Programming classifier with Naive Bayes classifier for filtering junk email. Katirai found that the Genetic Programming classifier was comparable with the Naive Bayes classifier in terms of Precision but was slightly out performed in terms of Recall. On its best run overall the Genetic Programming classifier achieved a precision of 95% and a recall of 70%, while the Bayes classifier had precision 96% and recall 77%. Katirai also discusses how email signatures can be harvested for useful features.

- ***Keyword verse naive Bayes* by Androutsopoulos et al** [1]. Androutsopoulos et al compare Naive Bayes for anti spam filtering with a keyword approach. The keyword approach uses the keyword patterns in the anti-spam filter of Microsoft Outlook 2000 (which Androutsopoulos et al believe to be hand constructed). There are 58 patterns that look for particular keywords in both the header and body of the email. Androutsopoulos et al found that the Naive Bayes approach out-performed the keyword approach, but the Keyword approach nonetheless had very good performance in terms of its precision which was 95%. The Naive Bayes approach had a precision of 98%, and its recall at 78% greatly outperforms the recall of the keyword approach 53%. In an earlier paper[2] Androutsopoulos was involved

[2]These collections are either based on the action that is performed on a message (task based features) or they are based on some previous partitioning (source based features).

with a memory based learning approach IiMBL (a simple variation of the K-Nearest Neighbour approach) was also evaluated for spam filtering on a different test corpus. The memory based approach was found to have similar performance to a Naive Bayes classifier.

- ***Naive Bayes* by Provost [15].** Provost compares the naive Bayes (with bag valued features) and RIPPER techniques for general email classification and spam filtering. He found naive Bayes significantly outperformed RIPPER in both cases. On the general classification problem the naive Bayes approach had 87% accuracy and RIPPER had 78%. For spam filtering naive Bayes had 95% accuracy after only learning on 50 emails while RIPPER didn't reach 90% accuracy till after learning on 400 emails.

2.3 Other

- ***Profile Usability*, by Pazzani [14].** Pazzani reported on a usability study where users were asked to assess their preferences for different approaches for representing email filtering rules. The representations compared are: key-word spotting, perceptrons, and prototypes[3]. A set of messages are labelled as either "*discard*" or "*forward*". Then the subjects learnt how to classify these messages. This was achieved by providing unlabelled examples to the subject which they attempt to "guess". Feedback was provided and the subject's performance improved. The subjects were then asked to rate different representations of email filtering rules. This was in terms of their willingness to use these rules for performing the same decision they had just learnt. The results showed that subjects preferred the prototype representation. Also the study found that word pairs, as opposed to individual words, were preferred by the subjects. This is a significant study as it considers the user's preferences in representing mail filtering rules.

2.4 Summary

Analytically comparing the approaches taken by different authors is very difficult, particularly since most tests are done on a very small number of email stores (often only one). Naive Bayes however looks like a good option for spam filtering with all authors reporting a precision of around 95%.

Although these quite high performance levels were achieved for two-category spam filtering, the more general categorisation task achieves far poorer results. This is unsurprising: useful email classification involves a user defining the class or classes within which they want to store a piece of email and this is a far less well defined task than distinguishing spam. When users make these classifications, there are many complex issues which define the process. For example, some users classify mail on the basis of the time by which they need to act upon it. Some classify mail according to the broad subject area as it relates to their work. We return to these issues in the next two sections.

3 Outstanding Problems

Some important aspects of the email classification domain are strikingly different from classical text categorisation and information retrieval. The most important differences are due to our goal of supporting individual users by predicting the classifications that they would make for their email.

It follows that learning needs to operate quickly, giving useful results on small amounts of data. It may also need to deal with concept drift, where the user's classifications change over time with their changing environment. There will also be considerable differences between users so that one user's classification of an email item will be different from another user's classification of the same mail item. Some of these may be inherently much harder to learn than others. Moreover, it is unclear how to judge success: what precision is needed for a classifier to be useful?

Another important aspect of the email domain relates to the user's needs in terms of control and understanding of their mail management. The classification processes should be scrutable: the user should be able to determine how it works and should be able to control each element of the processes which manage the classification.

4 i-ems approach to learning mail classes

A direct application of a classifier would be to automatically sort messages into their archive folders according the learnt classification. This is problematic for two main reason. First, users often wish to see all the incoming messages and if messages are automatically placed in folders then important message may be lost. Second, work is created for the user when the classifier incorrectly archives a message.

[3]The prototype represents email filtering rules by using two lists of words, say l_1 and l_2. The classification of a new message m is determined by which of the lists l_1 and l_2 contains more common words with the message m. Word pairs are also permitted within these word lists. A simple example is the follow rule for filtering junk email: *If there is more of "money", "save", "free" than "meeting", "deadline", "job done" then the message is junk otherwise it is okay.*

Segal and Kephart [19] explored one solution to these problems in MailCat. They provided the user with 3 buttons labelled with best mail folders to archive a messaged within. Hence a user simply clicked on the appropriate button, out of the three given, to archive the message. This increased the probability that the correct folder would be suggested. Also, the cost of an incorrect classification was minimal.

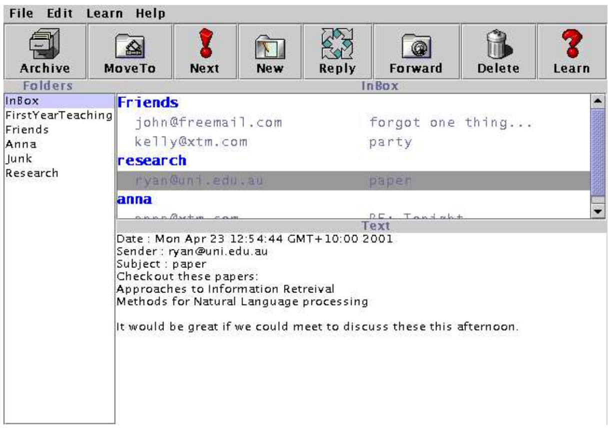
The approach we have take to address these problems also does not to automatically archive messages. Rather we sort each message within the *inbox* according to its predicted classification. Figure 1 shows an example of a i-ems screen. The long left panel lists the user's folders, which define the classification categories. To the right of this, there are two parts to the screen. The upper part shows the messages in the folder that is currently selected. In the figure, this is the inbox. This shows each message under its predicted folder category. For example, the top two mail messages have been predicted to belong in the *Friends* folder. Each appears as a single line with the sender and the subject. The user has currently selected the inbox single mail item classified under *research*. This is displayed in the lower right-hand window.

Once a user has read a message, there are two possible courses of action. If they are happy with the classification, they can simply click on *Archive* button. This is at the top left of the screen. This moves the message into that folder. In the case of the current message shown in Figure 1, the *Archive* button would move it to the *research* folder.

The other possible case is that the user is not happy with the classification. In that case, the user simply selects the *MoveTo* button followed by the name of the folder in the left panel. This moves the message to the correct folder. This is much the same amount of effort required for a user to archive messages in a standard email manager.

This interface should reduce the cognitive drudgery and the time taken to archive each mail message. If the system makes the correct classification, the user simply accepts that with a single click. If the system is wrong, the user does the sort of classification task they would have had to do anyway. This should significantly reduce the cost of incorrect classification while improving the overall interaction with the email manager.

The user may also scrutinise why a message is classified in a particular way by having the classification rules displayed. The form of this information is dependent on the type of learner used. For example the "dtree" learner displays a decision tree, whereas, the "keyword" learner displays a set of clauses. An example of the output for the "keyword" learner is shown in Figure 2. For example, the first of these means that a mail item is classified as belonging to the folder *Anna* if its sender is *anna*. The current form of these is arguably less obvious than the pseudo-English format that we could easily generate to appear like the text in the last sentence.

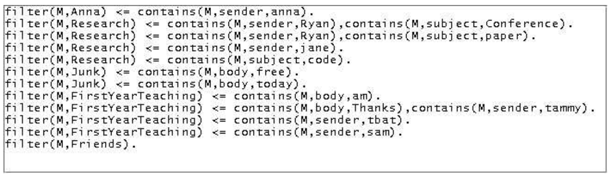


Friends
john@freemail.com forgot one thing...
kelly@xtm.com party
research
ryan@uni.edu.au paper
anna

Date : Mon Apr 23 12:54:44 GMT+10:00 2001
Sender : ryan@uni.edu.au
Subject : paper
Checkout these papers:
Approaches to Information Retreival
Methods for Natural Language processing

It would be great if we could meet to discuss these this afternoon.

Figure 1: A screen shot of the i-ems interface.



```
filter(M,Anna) <= contains(M,sender,anna).
filter(M,Research) <= contains(M,sender,Ryan),contains(M,subject,Conference).
filter(M,Research) <= contains(M,sender,Ryan),contains(M,subject,paper).
filter(M,Research) <= contains(M,sender,jane).
filter(M,Research) <= contains(M,subject,code).
filter(M,Junk) <= contains(M,body,free).
filter(M,Junk) <= contains(M,body,today).
filter(M,FirstYearTeaching) <= contains(M,body,am).
filter(M,FirstYearTeaching) <= contains(M,body,Thanks),contains(M,sender,tammy).
filter(M,FirstYearTeaching) <= contains(M,sender,tbat).
filter(M,FirstYearTeaching) <= contains(M,sender,sam).
filter(M,Friends).
```

Figure 2: An example set of rules generated by i-ems using the keyword learning approach.

5 Empirical Results

We now describe our experiments which compare the effectiveness of different learning approaches. We describe these briefly, then the way that we have tested them and the results achieved. Since our goal is to explore how well different approaches perform at the same time as their potential for scrutability, we then discuss these issues.

5.1 Learners Considered

The i-ems system uses an abstract class that permits a variety of learning approaches to be considered. Currently we have implemented four different approaches:

- **Sender** : This learns a set of rules that determine which folder to place a new message in based solely on the sender. For each sender a rule is created that filters new messages into the folder that the sender most commonly goes into. At the outset, we realised that this approach is limited in a variety of ways. For example a user may place messages originating from a single sender into different folders depending on the nature of their message. Also, a sender may have not sent a message to the user before, in which case the messages location is simply "unknown". In spite of these limitations, we explored the approach to see how well it would perform compared with more complex ones. This is especially important in light of our goal of scrutability.

- **Keyword** The keyword approach induces a set of clauses in a similar way to Quinlan and Cameron-Jones' FOIL [5]. The literals considered are whether a particular word is contained in one of the : sender, subject, or body fields. In a similar way to Cohen's Ripper [6] this learner starts to induce rules for the smallest folder and then progresses to larger ones.
- **TF-IDF** This approaches is an incremental learner maintaining a table of word frequencies as messages arrive. For details the reader is directed to Segal and Kephart's paper [19].
- **DTree** This is a simple decision tree learner. The d-tree constructed uses the information gain metric of ID3 [16]. This is done using 60% of the training data and the remaining 40% of the training data is used for pruning the tree.

5.2 Testing

We have implemented a 'Sliding Window' tester to evaluate the learning algorithms described. Given a list of emails which are chronologically ordered, the user both sets the number of messages in the test window and the number of messages the window is shifted along by in each iteration. The tester learns from the initial emails, and then tests on the following window of messages. This process is repeated as the training set is increased and the test window is shifted. This testing approach is advantageous as it reflects the way in which a learner that is integrated into a mail client would most likely operate. Also this approach gives a clear indication of how classification accuracy changes as the number of emails in the training set increases. Note that cross validation is inappropriate given the temporal nature of email.

We have conducted a preliminary evaluation of the four learning approaches using the above tester on a corpus of classified email belonging to one of the authors. The corpus contains 525 mail items in total and is classified into 6 different folders. The testing window contains 40 messages and is moved in steps of 10 messages. The accuracy is calculated as the percentage of the messages within the testing window that are correctly classified. These graphs are shown in Figures 3 to 6. [4]

The current implementations of "TF-IDF", "DTree", and "Keyword" learners always classify a new message into one of the existing folders. Hence, when reporting on the performance over all the folders it is pointless distinguishing between precision and recall as no messages are classified '*unknown*'. Whereas, in the case of the "Sender" learning approach, a message from any previously unseen sender will be classified '*unknown*'. Hence for this graph we have also graphed the precision, where precision is defined as the percentage of classified messages that are classified correctly.

Figure 3 shows that the TF-IDF approach although having quite good accuracy on a small

[4]The error bars indicate 90% confidence intervals.

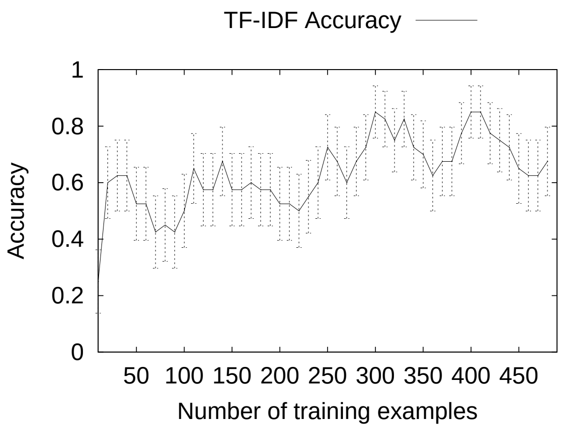


Figure 3: Graph of Accuracy when the TF-IDF approach is used.

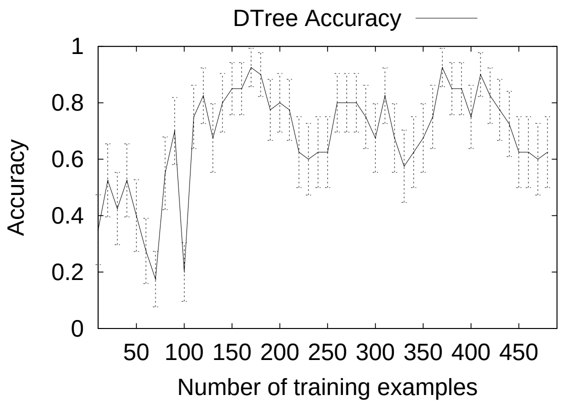


Figure 4: Graph of Accuracy when the DTree approach is used.

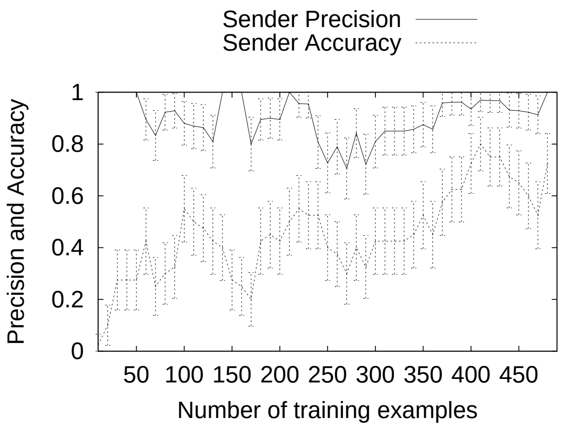


Figure 5: Graph of Precision and Recall when the Sender approach is used.

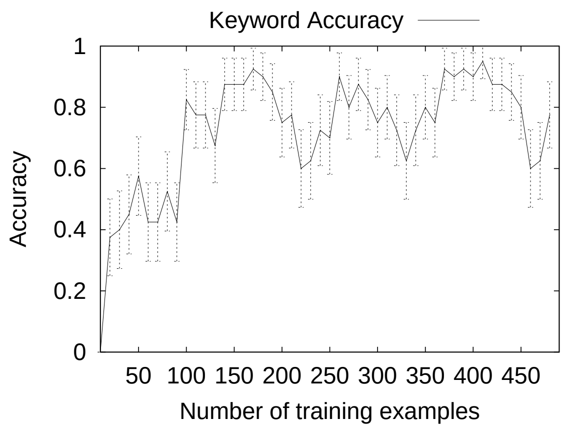


Figure 6: Graph of Accuracy when the Keyword approach is used.

number of emails, never achieves precision or recall much over 80%.

The DTree approach has a peak performance of around 90% which is better than the TF-IDF approach. However, as can be seen by comparing Figure 4 with Figure 3 its performance when the training set is small has some troughs at 20% and, importantly, it does worse for training sets below 75 examples.

The Sender approach, as shown in Figure 5, has excellent precision, but lower overall accuracy in comparison to the other approaches. For this domain, where precision will generally be more important than recall, this very simple method appears to perform quite well.

Overall the Keyword approach exhibited the best performance. This approach is also much more scrutable than either the DTree or TF-IDF. It seems to have similar performance to that of TF-IDF and DTree on small training sets but then is able to improve with larger training sets.

A summary of the average accuracy results is given in Table 1. As we have noted, the Sender approach appears to compare poorly due to its poorer recall. We also emphasise that these averages take account of results over the full range of training set size.

Learner	Accuracy
TF-IDF	68%
DTree	63%
Sender	45%
Keyword	72%

Table 1: The average accuracy of the different learning approaches.

6 Discussion and Conclusions

We have described the i-ems project and a range of related work in automated support for users in classifying their email. From the past work, we see that there has been considerable success in the classification of junk email. There has been rather less success on general classification. We believe that the middle ground needs to be explored. It seems likely that there are some mail classifications which are relatively easy for an automated tool to learn, with high precision. We also believe that there are categories which can be learnt with high precision and using very simple strategies. We expect that hybrid machine learning approaches will give useful results.

This paper has described our work to explore these issues. We have, indeed, found that the very simple Sender approach achieved high precision.

The results we have reported involve training set sizes with up to 500 examples. In this domain, it seems desirable that a learner should be able to give useful results with relatively smaller sets of training examples. It seems likely that most users will have relatively small numbers of mail items in each folder. So the results for small training sets are of considerable interest. Our results give accuracies around 50% for TF-IDF, DTree and Keyword approaches.

At the same time as we note the importance of assessing the learning algorithms on small training sets, it should be noted that the user has most to gain by automation in the handling of mail categories where they gets the largest numbers of mail items. Large numbers of training examples may be available for some folders for some users. So the results for larger training sets are still of interest.

We have already mentioned that one of the weaknesses of work in this area relates to the need for testing over many users. This paper reports experiments involving just one user. We are currently collecting data to enable us to repeat the experiments with larger numbers of users.

Another shortcoming in work in this area relates to the lack of standard test sets. Existing data sets like the Reuters collection may not be relevant to the way that individual users each manage the mail that they receive from a range of sources. We would like to develop a realistic data set that can be made widely available. Unfortunately, it is difficult to construct this. We cannot use authentic mail: even if the recipient user agreed to make their mail available, it is impractical to gain approval from all the senders of that mail. We are currently exploring ways to achieve the difficult task of anonymising authentic mail stores so that neither the sender nor the receiver can be identified.

We have also presented the i-ems interface which we are using to explore how to integrate automatic classification into a useful and usable interface. This is designed to take account of the limitations of automatic classification.

The i-ems domain is email classification and filtering. This domain has many of the elements that are common to emerging areas for personalisation. It should improve our understanding of the potential for a range of approaches to building individualised and automated text classification that users can readily understand and control.

References

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos and C. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anit-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167, 2000.

[2] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In *Proceedings of the Machine Learning and Textual Information Access Workshop of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD*, 2000.

[3] G. Boone. Concept features in re:agent, an intelligent email agent. In *Second International Conference on Autonomous Agents*, may 1998.

[4] C. Brutlag, J. Meek. Challenges of the email domain for text classification. In *Seventeenth International Conference on Machine Learning*, July 2000.

[5] R.M. Cameron-Jones and J.R. Quinlan. Efficient top-down induction of logic programs. *SIGART Bulletin*, Volume 5, Number 1, pages 33–42, 1994.

[6] W. Cohen. Learning rules that classify e-mail. In *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, pages 18–25, 1996.

[7] Lorrie Faith Cranor and Brian A. LaMacchia. Spam! *Communications of the ACM*, Volume 41, Number 9, pages 74–83, 1998.

[8] Eran Gabber, Markus Jakobsson, Yossi Matias and Alain J. Mayer. Curbing junk e-mail via secure classification. In *Financial Cryptography*, pages 198–213, 1998.

[9] H. Katirai. Filtering junk e-mail: A performance comparison between genetic programming & naive bayes, 1999.

[10] D. Kristol, E. Gabber, P. Gibbons, Y. Matias and A. Mayer. Design and implementation of the lucent personalized web assistant, 1998.

[11] W.E. Mackay. Triggers and barriers to customizing software. In *CHI'91 Conference on Human Factors in Computing Systems*, pages 153–160, New Orleans, Louisiana, 1991.

[12] P. Pantel and D. Lin. Spamcop: A spam classification & organization program. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, pages 95–98, 1998.

[13] T Payne and P. Edwards. Interface agents that learn: An investigation of learning issu es in a mail agent interface. *Applied Artificial Intelligence*, Volume 11, pages 1–32, 1997.

[14] M. Pazzani. Representation of electronic mail filtering profiles: A user study. In *Proc. ACM Conf. Intelligent User Interfaces.* ACM Press, 2000.

[15] J Provost. Naive-bayes vs. rule-learning in classification of email, 1999.

[16] J.R. Quinlan. Induction of decision trees. *Machine Learning*, Volume 1, Number 1, pages 81–106, 1986.

[17] J. Rennie. ifile: An application of machine learning to e-mail filtering. In *KDD-2000 Text Mining Workshop, Boston*, 2000.

[18] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[19] R. Segal and M. Kephart. Mailcat: An intelligent assistant for organizing e-mail. In *Proceedings of the Third International Conference on Autonomou s Agents*, pages 276–282, Seattle, WA, 1999.

[20] Steve Whittaker and Candace L. Sidner. Email overload: Exploring personal information management of email. In *CHI*, pages 276–283, 1996.