IEMS - an approach that combines hand crafted rules with learnt instance based rules

Eric McCreath ERICM@CS.ANU.EDU.AU
Department of Computer Science, The Australian National University, ACT 0200 Australia

Judy Kay JUDY@IT.USYD.EDU.AU
School of Information Technologies, The University of Sydney, NSW 2006 Australia

Elisabeth Crawford EHC+@CS.CMU.EDU
Computer Science Department, Carnegie Mellon University, Pittsburgh, USA

Abstract

Sorting email messages into folders is an important task for a vast number of computer users. The i-ems (Intelligent-Electronic Mail Sorter) project is focused on improving the way users interact with email managers. In this article we report on two aspects of the i-ems project. Firstly, we discuss how combining learnt and hand crafted rules can improve the overall performance of the system. Secondly, we present a composite rule learner that classifies mail by combining an instance based approach with an approach that constructs a general explicit description. We show that this approach improves classification performance. Furthermore, this combined approach produces understandable and concise classification rules that users can scrutinize allowing them to maintain a sense of control.

1. Introduction

The email manager has become a focal point for the way many people organise and conduct their activities (Ducheneaut & Bellotti, 2001). The i-ems (Intelligent Email Sorter) project has a broad aim of improving the tools that assist people in managing their email. The email application is a repository of often important documents that must be archived for possible later reference. Yet mixed with these important documents is a vast number of messages that are either: spam, mailing list messages that are not of interest, or messages that are only read once and dealt with. So helping users to effectively and easily deal with this deluge of information is both an important and difficult task. The i-ems project addresses this problem from two different angles. Firstly, we look at ways of learning rules that aid in the sorting of messages. Secondly, we focus on improving user interface design to enable a manager to incorporate learnt information while minimising negative side effects.

Different users archive messages in a number of very different ways. Some users do little archiving. Rather, they leave most of their messages in the inbox and then make use of search tools when retrieving previous messages. Others will archive messages based on categories from different aspects of their work/life (e.g. **teaching**, **admin**, **friends**, etc). By contrast, others will use more functional categories to organise their messages (**require reply**, **no action required**, **action this month**, **action this week**, **read and delete**, etc). These differences provide an opportunity for machine learning to help an email manager adapt to a particular user.

Standard email managers, such as Netscape and Microsoft Explorer, provide the user with the ability to add filtering rules to help users deal with messages that arrive. These rules are based on matching keywords within various fields of the received message. Filtering rules can be difficult and time consuming to construct. Also they may result in unforeseen and unwanted side effects. Mackay (Mackay, 1991) found that people avoid such customisation. Ducheneaut and Bellotti (Ducheneaut & Bellotti, 2001) state "Most of our users (17 interviewees, or 60 percent) say they don't use filters. Several simply haven't figured out how to use them, suggesting that either filters need to be simpler to use or that they are not that useful.". Our work flows out of addressing these concerns while aiming to improve the effectiveness and ease of handling

the deluge of messages.

In this article we describe the exploration of several approaches for the automatic induction of email filtering rules. Also, we look at combining hand crafted rules with learnt rules and show how this can be incorporated within a user interface. We describe work related to our approach in Section 2 and introduce the i-ems interface in Section 3. In Sections 4 and 5 we detail our approaches to filtering email. In particular, in Section 4, we explain how learnt and handcrafted rules can be combined and in Section 5, we show how instance based approaches can be incorporated into rules. We present a detailed evaluation of these approaches in Section 6. Finally in Sections 7 and 8 we conclude with a discussion of the results.

2. Related Work

A variety of approaches have been taken to address the problem of automating email classification. Most of these can be split into two groups: filtering junk email; and general classification of email. At first glance, one might presume email classification was simply a special case of text categorisation. However, even the seemingly similar work on the Reuters-21578 dataset is quite different from learning how to predict *individual* users' classification of their own mail.

In particular, it is important that learning algorithms produce useful results quickly, using only small amounts of data. An email user may have folders that contain only a small number of emails. Users also create new folders over time and hence it is important that classifications can be predicted based on only a small number of examples.

An important difference between the email classification problem and traditional text categorization is that the classification task may change with time. Changes in classifications might be due to changes in the user's activity: for example, a user who teaches a course in programming in one semester may not be involved in that type of activity in the next semester. This affects the task of a learner since it needs to recognise such changes. There are many other changes that affect classifications. For example, if the user's supervisor changes, or other personnel at work change their roles, a learner may need to adjust its classifications.

We note two other important aspects of this domain: user differences and differences in the difficulty in learning to predict the categorisation for different mail classes. We know that different people use quite different mail management strategies, as noted, for example in (Ducheneaut & Bellotti, 2001). We would expect that it is easier to learn the classifications applied by some users than would be the case for others.

Machine Learning and Information Retrieval approaches have demonstrated good performance can be achieved on spam/junk email. For example, *SpamCop* (Pantel & Lin, 1998), using a Naïve Bayes approach achieved accuracy of 94%. Sahami et al. (1998) applied a Bayesian approach and achieved precision of 97.1% on junk and 87.7% on legitimate mail and recall of 94.3% on junk and 93.4% on legitimate mail. Katirai (1999) used a genetic classifier and its best run overall achieved a precision of 95% and a recall of 70%. Androutsopoulos et al. (2000a) compared Naïve Bayes with a keyword approach. The keyword approach uses the keyword patterns in the anti-spam filter of Microsoft Outlook 2000 (which they believe to be hand constructed). They reported that the Keyword approach achieved precision of 95% and the Naïve Bayes approach 98%. On recall the corresponding performance was 53% and 78%. Androutsopoulos et al. (2000b) also explored a memory based learning approach IiMBL (a simple variation of K-Nearest Neighbour) showing similar performance to a Naïve Bayes classifier. Provost (1999) also evaluated Naïve Bayes, comparing it with the RIPPER algorithm, showing 95% accuracy after learning on just 50 emails while RIPPER reached 90% accuracy only after learning on 400 emails.

It is unclear quite how to compare this range of work since most tests are done on a very small number of email stores (often only one). However, Naïve Bayes seems to have achieved precision of around 95% in several studies. *Re:Agent* (Boone, 1998) also explored learning in a two-class setting. Boone used a hybrid approach with TF-IDF to learn useful features and then both neural networks and nearest neighbour approaches to distinguish 'work' emails form 'other' emails. This gave 98% accuracy on a dataset where the standard IR approach had 91% accuracy.

Although quite high performance levels were achieved for two-category spam filtering, results for the general classification task have been weaker. One quite strong result is described by Cohen (1996), who used the RIPPER learning algorithm to induce rules and reported 87%- 94% accuracy. He also explored TF-IDF, a classic Information Retrieval approach, and achieved 85%-94%. Cohen observed that the rule based approach provided a more understandable description of the email filter.

The *iFile* classifier (Rennie, 2000) was made available to several users to test on their own mail. The Naïve Bayes algorithm used by *ifile* resulted in 89% accuracy.

Support Vector Machines (SVMs) have proved quite effective for text categorization and Brutlag (2000) showed they also perform well for email. In a study using 5 email stores they found Linear SVMs achieved accuracies of between 70% and 90%. They compared this against the Unigram Language Model, 65% to 90%, and TF-IDF, 67% to 95%, depending on the store of email used.

The poorer results for general classification are unsurprising: useful email classification involves users 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 that 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. In fact, the results summarised seem very high for any realistic scenario where users might have a modest number of mail items in many of their mail folders.

While the work we have described suggests that automated classification should be able to operate usefully in helping users classify their email it is very difficult to compare the studies. In each case different data was used and since styles of email classification vary widely it is hard to know how well these results generalise. Recently, various researchers including (Klimt & Yang, 2004) and (Bekkerman et al., 2004) have promoted the use of the Enron email corpus as a standard test bed. The Enron corpus consists of the email messages of about 150 Enron employees. The Federal Energy and Regulatory Commission made the emails public during its investigation of the company. As yet, there have only been a few studies completed on the corpus. In particular (Klimt & Yang, 2004) have shown SVMs achieve a micro-averaged F1 of almost 0.7 and (Bekkerman et al., 2004) have demonstrated that a version of Winnow performs similarly to SVMs.

The work of (Brutlag, 2000) demonstrates that the performance of classification algorithms for some users is better than for others, and that accuracies on different folders for the same user can also vary. As such, it seems fruitful to explore approaches that can learn at least some classifications quickly. Even more importantly, it seems likely that a useful learner should be able to tell the user how well it performs so the user can decide whether that level of performance is good enough.

At the same time, since several approaches seem to achieve good results in some studies, we can afford to explore the usefulness of a range of approaches that are simplest to explain to the user. Thus the user should be able to understand any proposed classification rule and maintain a sense of control. This is the direction taken by Pazzani (2000) who reported a study where users were asked to assess their preferences for different approaches for representing email filtering rules.

Previous work also suggests the need to build classification mechanisms into an interface which *proposes* classifications rather than automatically acting on the mail. For example, the work on *MailCat* (Segal & Kephart, 1999), using a TF-IDF approach initially had error rates of 20% to 40%. Since this was considered unacceptable, Segal and Kephart took a different approach: *MailCat* recommended its three best predicted folders so that the user could archive email to one of these with a single click. This improved performance to acceptable levels. Given that for some users, studies have shown that state-of-the-art learning algorithms have error rates as high as 30% it is particularly important to try and minimise the cost of misclassification to the user. It is also important that the email classifier is able to explain its decisions to the user - this is a pre-requisite for the user being able to correct the classifier and also helps the user maintain a sense of control.

3. I-EMS Approach to Incorporating Automatic Classification Into An Email Interface

We now describe the experimental user interface that we have been developing as part of the i-ems project. This illustrates the type of mail reader facilities that we envisage as part of the support for users in managing email where there is machine learning assistance for classifying messages.

Direct application of a classifier could be used to automatically sort messages into their archive folders. This would mirror the approach available in many existing mail clients which can use rules to route specified classes of messages directly into a folder other than the inbox. We avoid this for two main reasons. First, users often wish to see all the incoming mail: if messages are automatically placed in folders, important messages may be missed. Second, any errors the classifier made would burden the user with additional work. Indeed, we are aware that the task of automated email classification is so difficult that an interface incorporating it should be based on the assumption that some classification errors are likely and that users must be able to maintain a sense of control.

An interesting approach is taken in *MailCat* (Segal

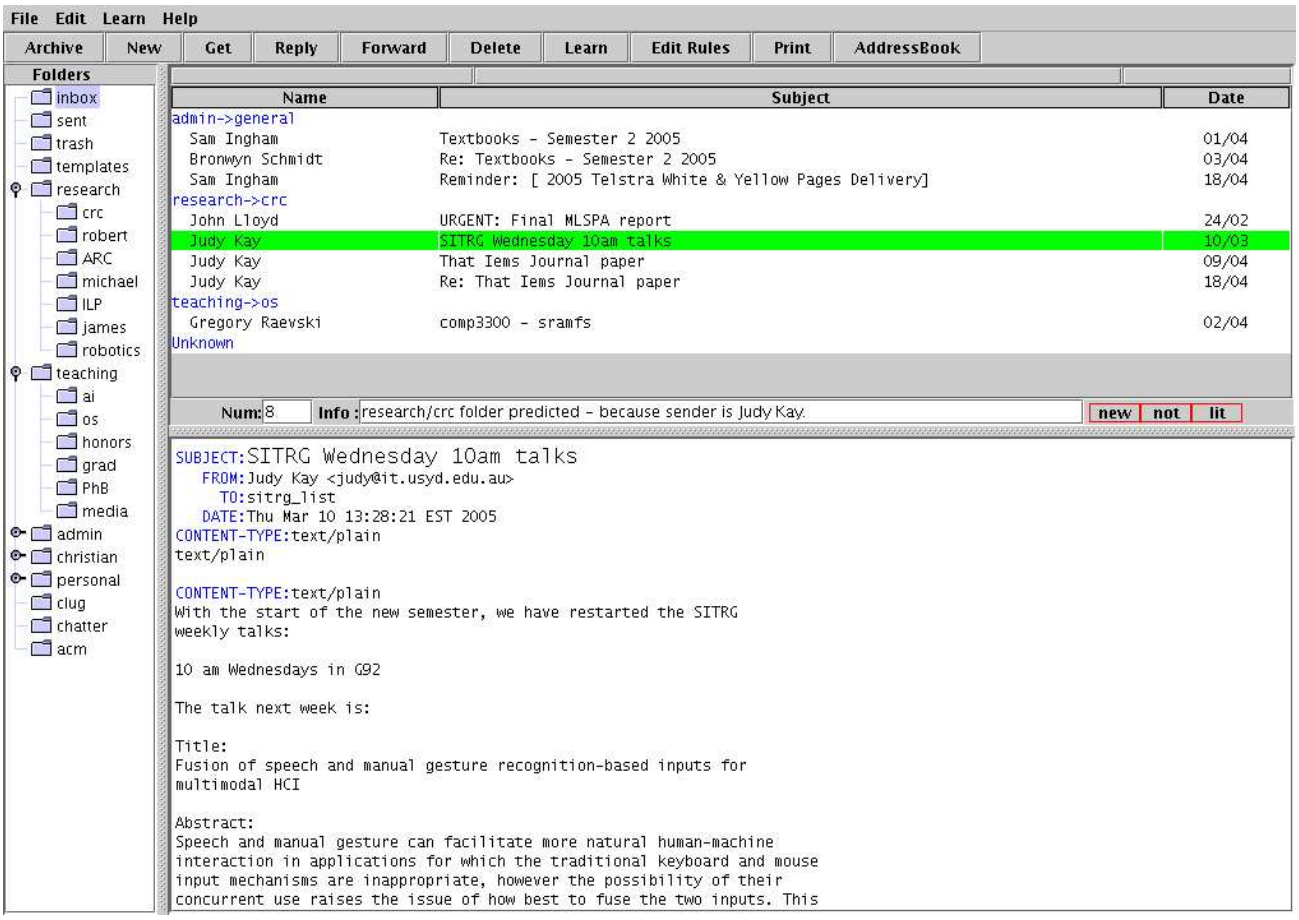

Figure 1. A screen shot of the i-ems interface.

& Kephart, 1999). As previously mentioned, Segal and Kephart created an interface with three buttons labelled with the best predicted mail folders for the current message. This increased the probability of offering a correct folder and also ensured the cost of an incorrect classification was minimal. We have taken an approach which is similar in spirit.

The i-ems approach sorts each message in the *inbox* according to its predicted classification. Figure 1 shows an example of an i-ems screen. The long left panel lists the user's folders. To the right of this, there are two main parts to the screen. The upper part shows the message in the folder that is currently selected. In Figure 1, this is the inbox, as indicated by the highlighting of its name in the panel at the left. The example shows how each message is displayed under its predicted folder category or under the category *Unknown* if no prediction has be made (in this example this category is empty). For example, the messages from Judy Kay and John Lloyd have been predicted to belong in the *CRC* folder. Each message appears as a single line with the sender and the subject. The user has currently selected a message from Judy Kay. The content of that message is displayed in the lower right-hand window.

Once a user has read a message, there are two possible actions. If the user is happy with the classification, he/she can simply click on the *Archive* button at the top left of the screen. In the case of the current message shown in Figure 1, the *Archive* button would move it to the *CRC* folder.

The other possible case is that the user is not happy with the classification. Then he/she can simply drag and drop the message into the correct folder. The work to do this is similar to that required to archive messages in a standard email manager. So, the user of our interface never has *more* work to do than moving the mail item to the correct folder. Once the user has moved any incorrectly classified items, a single click on the *Archive* button moves the rest of the messages to their correct folder. This approach should minimize the cost of incorrect classification to the user while improving the user's overall interaction with the email manager.

Ducheneaut and Bellotti (Ducheneaut & Bellotti, 2001) found that many users tend to organise email into folders around one or more of the following criteria: Sender, Organisation (e.g a client or professional body), Project or Personal Interests. Learning to classify email into folders based on the Sender is simple, but the other types of classification are much more difficult. Hence the accuracy of automated classification is likely to differ significantly between folders.

Another important aspect of our approach is that i-ems allows the user to *scrutinise* the classification process. The interface shows the user the reason an email was classified into a particular folder. When the user clicks on an email, it is displayed along with the reason for the classification. The explanation is displayed in the middle right hand panel (as seen in Figure 1). The learner that generates the hypothesis is also given the task of generating these explanations. In the cases when hypotheses are represented in clausal form, the clause that classified the message is simply conveyed as an English sentence. In Figure 1, this explanation indicates that the highlighted message was predicted to belong in the *research/CRC* folder because its sender was *Judy Kay*. This raises the important point of the scrutability of the explanations of different classifiers. We have implemented six different classifiers: Sender, Keyword, TF-IDF, DTree, Naïve Bayes, and Composite Rules. Each provides an explanation for its classification of individual messages. For example, the explanations generated by the *TF-IDF* learner simply contain a list of significant words.

This description of the i-ems interface and approach is important for this paper because it is the context for the work reported in the following sections. Our experiments in classifiers for email are intended to fit into an interface which ensures the user is able to scrutinise the operation and performance of the classifiers.

The final way in which the user can scrutinise the classifier is by asking to see the rules it uses to make classifications. The form of this information depends on the classifier being used. An example of the output of the *Keyword* learner is shown in Table 1. The first clause shown means that a mail item is classified as belonging to the folder *Anna* if its sender is *anna*. The current form of these clauses is arguably less obvious than the pseudo-English format we could easily generate to appear like the text in the last sentence.

Users may also view the classifier to scrutinise its classification. Table 1 shows a classifier generated by the *Keyword* learner.

The TF-IDF approach is limited in the accuracy that can be obtained as the user model induced does not reflect the way a user decides on the right folder for archiving an email item. Another problem with this approach is that the user is unable scrutinise why the agent placed an email item in a particular folder. There is a similar problem with a Naïve Bayes approach.

Table 1. An example set of rules generated by i-ems using the keyword learning approach.

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

Approaches such as key-word spotting, perceptrons, and prototypes provide an understandable profile of a user's email filtering rules. However, they tend to be "brittle" and require a large number of training examples to learn. This is due, at least in part, to the dimensionality of the space in which these systems must learn.

The goal of scrutability makes it attractive to explore approaches that use a richer language for describing profiling rules. This has the potential to provide a profile that is more scrutable. Also it has to potential to address some of the "brittleness" issues that flow from the dimensionality of the problem. We want to explore ways to build a profile which closely matches the way that real users model and filter messages. Our approach uses a restricted first order language for representing profiles and we explore a variety of ILP techniques.

4. Combining hand crafted rules and learning approaches

In existing email interfaces the user is able to add rules to filter email messages. We are interested in developing systems that learn rules to sort email messages into folders. This raises an important question 'How should these two approaches be combined?'. Clearly if the learner and the hand crafted rules are in agreement

on a message then this message can be simply classified with either approach. If one of the approaches is uncertain about how to classify a message, labelling it as 'unknown', then the other approach can be used. However, if the two approaches classify a message into different folders then a decision needs to be made. Given that we have found the handcrafted rules to be more precise than the learnt rules the hand crafted rules are more likely to correctly classify the message. Moreover, our goal of user control makes it logical to follow this policy. Hence the hand crafted rules are given precedence. In fact learnt and hand crafted rules complement each other nicely as the learnt rules can have better recall as they tend to make predictions on a greater percentage of messages and the hand crafted rules tend to have poorer recall yet have better precision. The utility of this approach is clearly seen in the results section of this article.

5. Composite rules: Using a richer hypothesis space

Instance based learning approaches, such as k-nearest neighbour, have a number of advantages over learners that explicitly form generalisations. Firstly, instance based learning approaches do not require retraining: the new labelled examples are simply collected and used when a prediction is required on an unlabelled instance. In this sense they are 'lazy' because the inductive generalisation is only made during a prediction and is never explicitly generated. Secondly, they can easily exploit locality as only the labels of the examples that are 'close' to the instance being predicted influence the classification of the instance. Note that, this limits the biases within a generalisation and possibly leads to over-fitting. This also means the 'hypothesis space'[1] may be considerably more complex than that used by a learner that explicitly forms a generalisation.

There are also some disadvantages with instance based approaches. Firstly, there is no real explanation 'why' the approach makes a particular prediction other than 'because it was like previous examples provided'. In contrast, a learner that explicitly generates a hypothesis can provide an explanation for a prediction in terms of the induced hypothesis: this can be presented to the user. Secondly, instance based approaches have scaling problems, as the time taken to make a prediction is generally linear with respect to the number of training examples. Learners that form explicit generalisation do not have these scaling problems – once the system has generated the induced hypothesis, the time to classify an unlabelled example is independent of the size the training data.[2]

We present an approach that generates an explicit hypothesis based on an instance based approach. It works by learning rules that direct the use of an instance based approach. These rules will be different for different users. However, they are inherently stable as they do not contain any explicit content from the mail messages. Rather, they direct the use of an instance based approach which examines the message content.

The learnt rules we generate are expressed in clausal form, where the clauses are generated in a top down fashion in a similar way to FOIL(Cameron-Jones & Quinlan, 1993). Also, as in FLIP(Bergadano & Gunetti, 1994), no negative examples are provided to the learner. Rather they are implied by the functional nature of the training set. The approach has some similarities to FOIDL(Mooney & Califf, 1995) which uses intensional background knowledge and arranges the clauses as a decision list. Our learning system does not use a greedy covering approach[3], but rather, it uses all the examples when inducing each clause. This is partly due to the fact that there are no negative examples: so the examples that place messages in a different folder are treated as negatives example. This approach was also adopted because it produced a more conservative set of clauses. That is, if the learner is uncertain of the classification of a mail item, then it can simply label it as 'unknown' so that it can be placed in under the label 'unknown' in the inbox.

The hypothesis space is best explained with an example. Table 2 shows the set of clauses generated from our first user's mail. The first clause in this table states the following:

> If the last five messages from the sender of the new message have been placed in the same folder F and F is not the 'ml' folder then classify the message as F.

Such rules are intuitive and they work effectively for different senders and different folders. Notice that the

[1] Although in instance based approaches there is no explicit hypothesis space since there is no hypotheses to enumerated, there is an implicit hypothesis space that is defined by the shape of the concepts the instance based approach can represent.

[2] However, the size of the hypothesis may be dependent on the size of the training data.

[3] A greedy covering approach learns by finding a clause that covers as many positive examples as possible adding it to the final hypothesis then removing these covered positive examples. This process is repeated until all the positive examples are explained.

nature of the rules means that even as the underlining data changes, the induced rules can be relatively stable. So, for example, if the actual folder categories change, this rule can still operate effectively. This rule works well for people who organise their email by consistently classifying mail from the one sender into the same folder. In our experiments, we have found that for some users this is the case.

Also, this type of rule copes well with the case where there are particular folders, such as 'ml' in the example, where the general rule should should not be applied. Even with a list of such exceptions, the rule is compact, simple and intuitive.

Note, however, that in the case where the user places mail from a particular sender into a number of different folders, this rule will fail. In that case, another rule would be required to make a prediction.

The second clause in Table 2 states the following:

> If there is a message in folder F with the same 'subject' as the unclassified message and F is not the 'colleagues' folder then classify the message as F.

This deals with the very common case where a series of messages are sent between users, retaining the same subject. Such messages sequences would be often archived in the same folder. Note that, the 'Re:' that often appears in the subject line is ignored. This class of rule is clearly important for the mail, as reflected in many mail interfaces that sort mail by subject line and the fact that the reply action on such mail interfaces preserves that subject line. Once again this rule operates over all folders and is essentially independent of the actual data in question.

The third clause in Table 2 states:

> If folder F has a message with a similar 'subject' to that of the unclassified message and F is not the 'csp', 'colleagues' or 'ml' folders then classify the message as F.

In this case, similarity is measured using a IDF[4] metric. This metric takes the intersection of words between the messages and the IDF weighting is summed. Self similarity is used to normalise this metric. If $idf(w)$ is the inverse document frequency of the word w, and W_1 and W_2 are multi-sets of words of the two mail items being compared, then the similarity of the second mail item to the first is:

$$sim(W_1, W_2) = \frac{\sum_{w \in W_1 \cap W2} idf(w)}{\sum_{w \in W_1} idf(w)}$$

In the example, the rule applies this to the subject of the mail. The same approach can be applied to the body of a message, as in the next example in the Table. A threshold is used to determine if two subjects (or bodies) are considered similar. Hence the literal :

contains_similar(F,M,subject,threshold,'idf table')

is true if there exists a message M' within the set of archived messages such that :

sim(M.subject, M'.subject) > threshold

Table 2. Clauses induced by the 'Composite Rules' learner with the first users messages.

filter(M,F)⇐lastXsender_placements(F,M,5), F ≠ ml.
filter(M,F)⇐contains_same(F,M,subject), F ≠ colleagues.
filter(M,F)⇐contains_similar(F,M,subject,0.6,idf table), F ≠ csp, F ≠ colleagues, F ≠ ml.
filter(M,F)⇐contains_similar(F,M,body,0.6,idf table), F ≠ colleagues, F ≠ dp.

The clauses generated are considerably more compact than those generated by approaches such as keyword spotting. Yet, they still provide a basis for explanation of the classification and so, they can be scrutinised by the user who wants to examine why the system operates as it does.

Table 3 shows the clauses learnt from the set of data generated from the second user. Note that this user archives their messages quite differently. They mainly classify them in terms of what they need to do with the message rather than the types of categories chosen by the first user. The second user's classification is a much more difficult for this, or any approach, to learn good classifications. As can been seen in the table, this user's mail categories include aspects relating to urgency, such as whether the mail should be dealt with this week.

The algorithm, shown in Algorithm 1, works by considering the performance of each of the four possible clauses. It then tunes these clauses by adding literals that exclude folders in which the clause does not perform well on. Finally poor performing clause are completely removed. The four possible clauses are

[4]IDF is the inverse document frequency, this gives lower values to frequent words and higher values to infrequent word.

constructed from the four seed literals, namely: "last sender placements", "contains same subject", "contains similar subject", and "contains similar body". These four clauses are seen in the hypothesis shown in Table 2. To tune each of these clauses the algorithm evaluates the clause on the entire set of messages against each of the folders. This evaluation calculates the number of message correctly and incorrectly placed in each of the folders. When determining the classification of the next message the clause may only make use of messages temporally before the message being classified. This approach is also used in the evaluation of the clauses. Hence the sorting of messages in chronological order. Once the table of correct/incorrect messages for each folder is constructed it is used to determine which folders to exclude. e.g. the "ml" folder is excluded from the first clause in the hypothesis in Table 2. The information collect about the performance of the clause is also used to determine which clauses to completely remove from the final hypothesis. For example the hypothesis in Table 3 has had both the "last sender placements" clause and the "contains similar body" clause remove as their overall performance was shown to by unfruitful.

Table 3. Clauses induced by the 'Composite Rules' learner with the second user messages. The large number of exception indicates that the recall will be poor.

filter(M,F)⇐contains_same(F,M,subject), F ≠ M_ReplynDelete, F ≠ This_week, ···, F ≠ Projects. (18 exceptions in total)
filter(M,F)⇐contains_similar(F,M,subject,0.6,idf), F ≠ M_ReplynDelete, F ≠ Junk, ···, F ≠ Projects. (25 exceptions in total)

6. Empirical Results

In this section we present our empirical study. We first briefly describe the different learners we use for comparison, and then we detail and motivate the approach used in conducting these experiments. Finally we present and discuss the results.

As we wish users to be able to scrutinise the reason for a particular classification we do not evaluate the different learning approaches solely by precision and recall. Rather, we also discuss how scrutable the induced hypothesis are for each of the learning approaches.

6.1 Learners Considered

We consider the following six different learning approaches:

chronologically sort T
$H := \{\}$
foreach $L \in$ four seed literals **do**
  $C :=$ "filter$(M, F) \Leftarrow L$"
  foreach $f \in$ folder labels **do**
    correct$_f := 0$
    incorrect$_f := 0$
  od
  foreach $m_i \in T$ **do**
    $f :=$ folder classification of C on m_i using
      messages $\{m_0, \ldots, m_{i-1}\}$
    if $f =$ the folder label of m_i **then**
      increment *correct*$_f$
    else
      increment *incorrect*$_f$
    fi
  od
  foreach $f \in$ folder labels **do**
    if incorrect$_f > 0.4 \times$ correct$_f$ **then**
      add "$F \neq f$" to the body of C
    fi
  od
  if C predicted more *correct* than *incorrect* **then**
    $H := H \cup C$
  fi
od
return H

Algorithm 1: Pseudo code of the composite rules algorithm. The algorithm is given n training messages with folder labels, this is denoted T. It returns a set of clauses H.

- **Sender** : The "sender" learning approach creates a list of rules where each rule corresponds to a *sender* in the training set. The sender is simply the 'from' part of the email. Each rule filters emails from a particular sender into the most common folder emails from that sender were placed into in the training set. For example if messages from 'Bob' mostly go into the 'Friends' folder then the following rule is created:
  filter(M,'Friends')⇐sender(M,'Bob').

  This simple approach has two main shortcomings. Firstly, a user may place emails from the same sender into different folders. For example Bob may be a colleague, as well as a friend, and hence some messages from Bob may belong in a 'Work' folder. Secondly, emails from new senders (senders that do not appear in the training set) will necessarily be marked as 'unknown'. Despite these shortcomings this simple approach performs quite well and hence forms a useful baseline for comparison with more complex approaches.

  The rules generated by this approach may be simply scrutinised by the user.

- **Keyword** : This approach views the sender, subject, and body fields of messages as a bag of words. The learning approach induces a set of clauses which have literals that test if a particular word is an element of one of the fields of the message. The approach we use works like Foil (Cameron-Jones & Quinlan, 1994), constructing clauses in a top-down fashion. Also in a similar way to Cohen's Ripper (Cohen, 1996) rules are constructed in ascending order of folder size(smallest folders first). The rules constructed by the keyword learner are easy for the user to scrutinize. Furthermore, any particular classification made can be explained by describing the rule that made that classification.

- **TF-IDF** : This approach maintains a table of word frequencies of the messages. Words that appear often in a particular folder, but are not common to all messages, are considered good indicators for a message being part of the folder. Each folder is thus associated with a large list of weighted words. Emails that require a prediction use the weighted list to determine the closest folder. The approach we used follows Segal and Kephart (Segal & Kephart, 1999). It is difficult for the user to scrutinise the hypothesis as the weighted lists are generally very large. However the most important words in the voting can be shown to the user.

- **DTree** : In this approach 60% of the training data is used to construct a decision tree that fits the data as well as possible (the tree is recursively constructed using the training data until either the leaf nodes contain training examples of a single class or no more splits are possible). The information gain metric of ID3 (Quinlan, 1986) is used to determine which attribute to split on. The remaining 40% of the training data is used for pruning the tree. A user can understand and scrutinize decision trees, although they can be cumbersome to display when large.

- **Naïve Bayes** : Our Naïve Bayes implementation is based on the algorithm in (Mitchell, 1997). We have increased the influence of the sender and the words in the subject of the email by increasing the weight of their conditional probabilities during classification. We found that this increased accuracy. A threshold value is used to determine when to classify an email as 'Unknown'. In a similar way to the TF-IDF approach the Naïve Bayes approach is not particularly scrutable.

- **Composite Rules** : This approach was described in Section 5.

6.2 Testing

We have created a testing approach that parallels the way a user engages with an email manager. At any point in time the email manager needs to be able to predict which folder to place incoming messages into, the manager may use training data that is currently archived in folders. Thus, our testing approach assumes messages arrive in chronological order with a testing window sliding across these messages. Also the training data consists of all the messages before the testing window. This testing approach gives both an estimate of the percentage of misclassification the user would experience and also a clear indication of how classification accuracy changes as the number of emails in the training set increases.

Cross validation would be an inappropriate testing approach in this domain. Firstly, in most cases the training email would arrive after testing emails, whereas, this would not happen is a real system. Secondly, at some point email is received that is for a new folder or is a type of message that has not been seen before. You would expect any system to be unable to make an accurate predication on such a message. However, if cross validation is used, there may always be some examples of such messages in the training set. This would artificially improve the measured performance of the learner.

Due to the private and often confidential nature of email it is difficult to collect authentic data. Our users provided classified email that they were comfortable giving us. This meant some of the messages were removed from their collection. However, we do not believe this has significantly perverted the findings of this study.

6.3 Comparing handcrafted and learnt rules

We now present results that shows how the combination of handcrafted and learnt rules improves the overall performance of the system. For this evaluation we use a large corpus of 5100 messages from a single user. These messages were collected over a period of 3 months. The user had 70 hand crafted rules and 21 different folders. The testing window contains 100 messages and was moved along in steps of 100 messages. The simple 'sender' learning approach was used in this comparison. The 'precision' is calculated as the percentage of correctly labelled messages that the approach makes a prediction on(ie that were not labelled 'Unknown') These results are shown in Figure 2. Figure 3 also shows the percentage of messages that are classified as 'Unknown' by the different approaches. Clearly the handcrafted rules had the most precision yet they made predictions on far fewer messages. Whereas the learnt rules had poorer precision yet made many more predictions. By combining both approaches the number of messages unknown is reduced while the precision is maintained.

These results are reflected in Table 4. The recall is the percentage of correctly classified examples from all the examples in the test set. By combining the learnt rules with the hand crafted rules the recall is better than either of the individual approaches. Generally we would like to increase both precision and recall. In many instances the improvement in one measure should not be at the cost of the other. Hence IR researchers often use the 'f1' measure which provides a sensible trade off between these two measures. The 'f1' measure is the harmonic mean of 'recall' and 'precision'[5]. The combined approach out performs either of the individual approaches in terms of the 'f1' measure.

6.4 Evaluation of the 6 learning approaches

We now present results from our evaluation of the six learning approaches using the sliding window testing approach on a corpus of 5 users. The users in this study have a variety of different approaches for determining which folder to archive their messages into.

[5] $f1 = \frac{2pr}{p+r}$ where p is the precision and r is the recall.

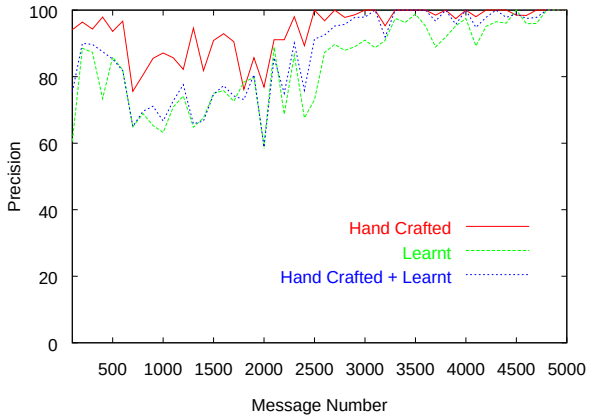

Figure 2. Sliding window precision results on the single user test.

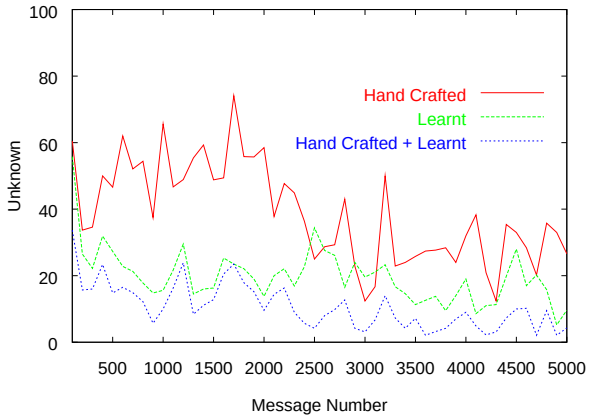

Figure 3. Sliding window 'unknown' results on the single user test.

User 1 archived messages simply based on the domain that the messages related to (i.e, administration, friends, teaching, etc), whereas User 4 archived messages based on the actions performed on the messages (i.e, read, replied to, deleted, requires response within a week, etc). Clearly the first of these users is easier to classify than the latter. This is reflected in the results.

In this evaluation we use a testing window that contains 30 messages and we slide it along by 30 messages in each run. The number of messages and folders for each of the users is shown in Table 6. The accuracy (or recall) is calculated as the percentage of messages in the test window that are correctly predicted by the classifier (Table 7). As the classifier may also label a message as 'Unknown' we tabulate the percentage of 'Unknown' classifications (Table 9). The precision is the percentage of message correctly classified that were given a non-unknown class label (Table 8). Finally the precision and recall is combined into the 'f1' measure to give an overall indication of performance

Table 4. Comparison of performance on the first half of the corpus. (HC = Hand crafted, LO = Learnt only, and HC+L = Hand crafted and learnt)

Approach	Precision	Unknown	Recall	F1
HC	85.0%	52.5%	40.4%	54.8%
LO	71.5%	29.1%	50.6%	59.3%
HC+L	71.9%	20.0%	57.5%	63.9%

(Table 10). Note that, boxes indicate row maximums.

A summary of the average recall/precision/f1 results is given in Table 5. Because of a few missing values for the D-Tree learner we have not included its results in this table.

Table 5. The average accuracy of the different learning approaches.

Learner	Recall	Precision	F1
Sender	46.8%	70.0%	55.6%
Keyword	43.8%	56.2%	48.1%
TF-IDF	43.3%	43.3%	43.3%
Naïve Bayes	31.3%	55.6%	39.5%
Composite Rules	40.0%	78.2%	49.1%

Table 6. Number of messages and folders for each of the users within the study.

User	#messages	#folders
1	526	7
2	949	35
3	869	12
4	429	9
5	1500	24

To give an indication of how the different learning approaches perform over time we have generated graphs for User 1. These graphs are shown in Figures 4 to 9. [6] These figures show accuracy/precision/unknown as the testing window slides across the training data. Note that, only accuracy is shown for the Keyword, TF-IDF, and D-Tree approach as the percentage of unknown examples is zero in these cases and hence the accuracy and precision are identical.

We have not presented a comparison of the times the different algorithms required for execution. However, to provide the reader with an idea of the running times each of the learners was run on all 526 messages of User 1. All the learners are implemented in Java and were run on a 1GHz AMD Duron producing : Sender 0.1 sec, Keyword 27 sec, TF-IDF 10 sec, Naïve Bayes 1 sec, and Composite Rules 6 sec.

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

Table 7. Recall of the different approaches. (S=Sender, K=Keyword, TI=TF-IDF, DT=DTree, NB=Naïve Bayes, CR=Composite Rules)

User	S	K	TI	DT	NB	CR
1	47.5	**69.4**	62.7	68.2	56.3	64.7
2	**41.3**	15.6	24.2	-	28.9	29.1
3	**70.4**	43.7	46.5	61.4	39.3	49.5
4	28.5	**41.3**	29.2	45.9	19.7	2.3
5	46.3	49.0	53.9	-	12.2	**54.4**

Table 8. Precision of the different approaches. (S=Sender, K=Keyword, TI=TF-IDF, DT=DTree, NB=Naïve Bayes, CR=Composite Rules)

User	S	K	TI	DT	NB	CR
1	**91.0**	69.4	62.7	68.2	78.2	87.8
2	59.0	35.2	24.2	-	54.0	**67.6**
3	80.4	61.3	46.5	61.4	64.2	**83.9**
4	51.6	43.5	29.2	45.9	43.3	**75.0**
5	67.8	71.8	53.9	-	38.2	**76.8**

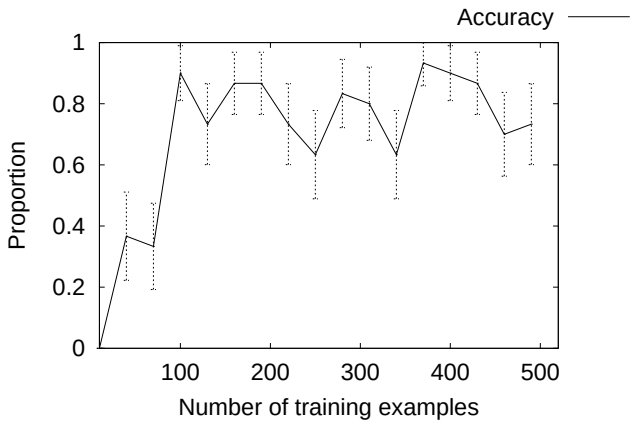

Figure 4. Graph of Accuracy for the Keyword Approach.

Figure 5 shows that the TF-IDF approach, although having quite good accuracy on a small number of emails, never achieves precision/recall much over 80%.

The DTree approach has peak performance of around 90% which is better than the TF-IDF approach. However, as can be seen by comparing Figure 6 with Figure 5 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 7, has excellent precision, but lower overall accuracy in compari-

Table 9. Percentage of unknown classification for the different approaches. (S=Sender, K=Keyword, TI=TF-IDF, DT=DTree, NB=Naïve Bayes, CR=Composite Rules)

User	S	K	TI	DT	NB	CR
1	47.5	0.0	0.0	0.0	28.0	26.3
2	30.0	55.7	0.0	-	46.5	56.9
3	12.5	28.8	0.0	0.0	38.8	41.0
4	44.9	5.1	0.0	0.0	54.4	96.9
5	31.7	31.6	0.0	-	68.2	29.1

Table 10. f1 measure for the different approaches. (S=Sender, K=Keyword, TI=TF-IDF, DT=DTree, NB=Naïve Bayes, CR=Composite Rules)

User	S	K	TI	DT	NB	CR
1	62.4	67.1	62.7	68.2	65.5	74.5
2	48.6	21.6	24.2	-	37.7	40.7
3	75.1	51.0	46.5	61.4	48.8	62.3
4	36.7	42.4	29.2	45.9	27.1	4.5
5	55.0	58.2	53.9	-	18.5	63.7

son 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 for this user.

The Naïve Bayes approach shown in Figure 8, achieves a precision of 100% at its peak, however the precision fluctuates and the overall accuracy never goes above 80%.

Overall the Composite Rules approach exhibited the best performance in terms of precision(See Table 8). This approach is also much more scrutable than the DTree, TF-IDF and Naïve Bayes approaches. It seems to have similar performance to these approaches on small training sets but then is able to improve with larger training sets.

7. Discussion

Clearly no one learning approach performs best for all the users. Also the approaches work very well for some users and very poorly for other users. This may simply mean that such technologies would only be useful for some users. It also provides a challenge to think how this limited technology may be applied most effectively.

There has been considerable success on the classification of SPAM, yet, rather less success on general classification of email. Focusing on user interface issues is most likely to produce the greatest gains in helping users deal with the deluge of information that is

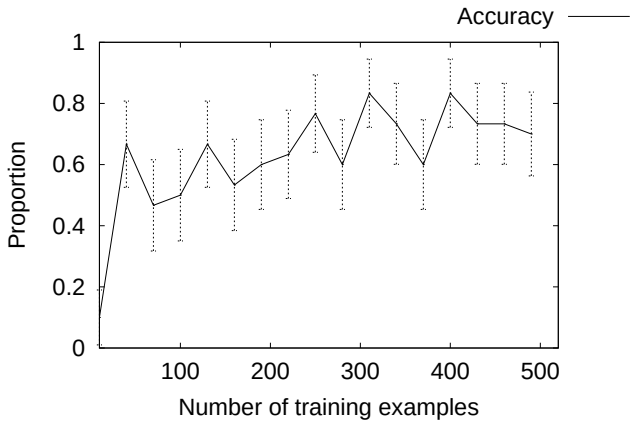

Figure 5. Graph of Accuracy for the TF-IDF Approach.

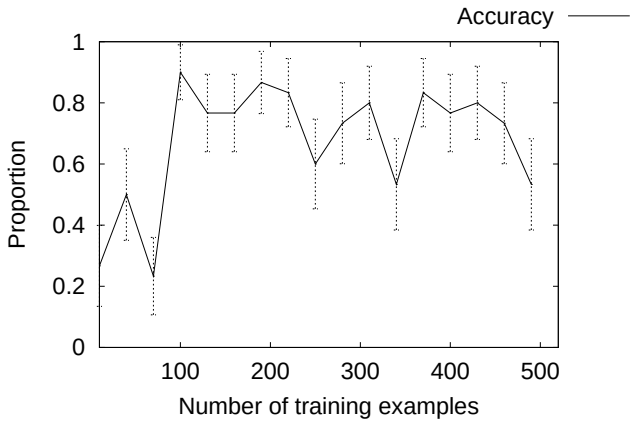

Figure 6. Graph of Accuracy for the DTree Approach.

sent via email. One possibility is to investigate obtaining a level of certainty for a particular classification, this would enable the email manager to make a better judgement about how to deal with a particular message. For example if the machine learning system is 99.99% certain that a message is SPAM then it could be deleted without showing it to the user, whereas, if it was say only 80% certain then it may place it somewhere in the inbox for the user to inspect.

It is worth briefly discussing the performance of the different approaches for each of the users, taking account of the qualitatively different character of the learning tasks across the users. Although we have reported F1 as the measure that combines precision and recall and F1 weights both of these equally, it is not clear that all users would choose this. Indeed, an interface that was able to show the user the trade-off associated with the different learners could then allow the user to decide which works best for them and their personal classification style.

User 1 had a small number of folders (7) for their 526 email items. This user archived messages based on cat-

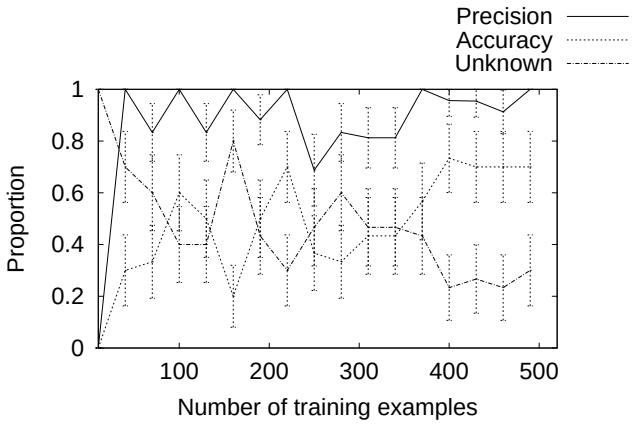

Figure 7. Graph of Precision, Accuracy and Unknown for the Sender Approach

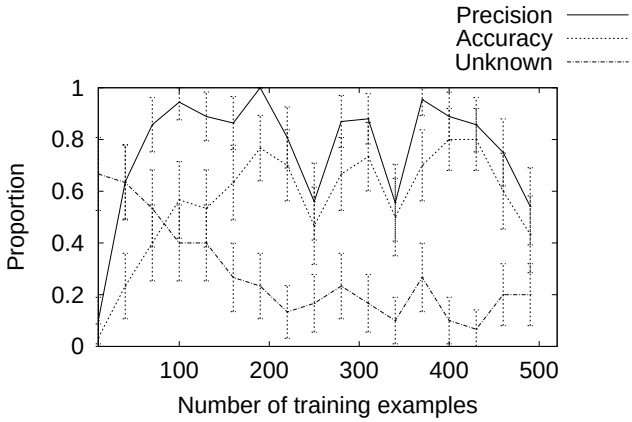

Figure 8. Graph of Precision, Accuracy and Unknown for the Naïve Bayes Approach

egories from different aspects of their work/life. This made the process of learning much easier and hence the overall good performance on all approaches. The Sender approach gave the highest precision (91%) for this user and for any user under any learner. Keyword spotting gave the best recall even though it avoided placing any items into the "unknown" category. The F1 measure favoured the Sender approach. However, this is a case where the particular user might be advised of the overall performance of these approaches and might decide which precision-recall trade-off they found most useful.

User 2 tended to classify email according to what they needed to do with it and when they needed to attend to it. So, for example, they had a folder for things to deal with this week and this folder had mail items associated many different tasks. This user also had the largest number of folders (35) for their 949 messages. This is clearly very difficult for any classifier. For this user, CR's good precision indicates that these rules may have captured the concept drift associated with

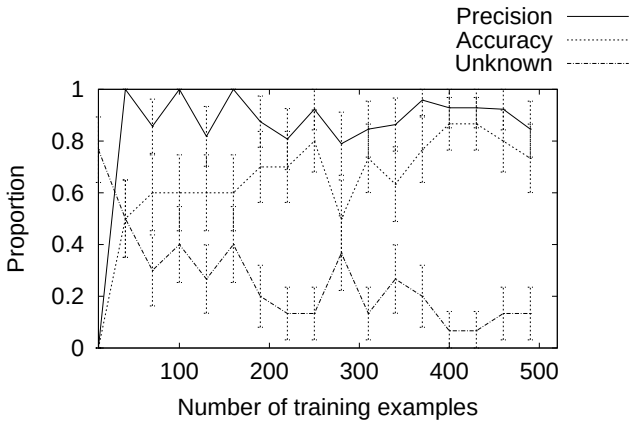

Figure 9. Graph of Precision, Accuracy and Unknown for the Composite Rules Approach

the user's current priorities.

User 3 classified messages in a similar way to User 1. The performance was also good across the learning approaches. Although, the approaches did not perform as well as they did with User 1. This is mainly due to the fact that User 3 had 12 folders rather than the 7 folders of User 1.

User 4 and User 1 had a similar number of messages and folders. However, all of the learning approaches performed poorly for User 4. This is due to the fact that User 4 classified messages purely using functional categories. The classifications included: **ToActOn**, **Keep**, **ReadAndKeep**, and **ReplyAndDelete**. The difference between these classification may not be revealed within the content of the messages, rather, a much broader context may be required to distinguish their classification. The composite rule approach classifies 96.9% of the messages as "unknown". Effectively it determined that any classification made would be no better than guessing, and hence, it simply labels the messages as "unknown". This high number of "unknown" classifications is reflect it the very low F1 measure for the composite rule approach. It could be argued that the composite rule approach is more helpful to the user, as at least the user knows that the learning approach is unable to offer good predictions. Note this also brings into question the utility of the F1 measure for ranking different approaches. Possibly a better way of ranking learning approaches would be to place a threshold on precision and then order the approaches based on recall. As a user may require a certain precision for an approach to be satisfactory, however, once this condition is meet then the highest recall is desirable. Note also, the other approach's performance is mainly due to them predicting the majority class(**ReadAndDelete** makes

up 48% of the examples).

User 5, like User 1 and 3, based their categorization on different aspects of their work/life. The performance for most approaches was generally good although not as good as User 1 and 3, this is mainly due to the larger number of folders considered(24 folders).

The composite rule learning approach performs well in terms of precision and recall. It also enables the user to easily scrutinise a classification made by the system. Also the combining of handcrafted and learnt rules clearly shows how the overall performance can be improved by this hybrid approach. This also opens up the possibility of having a number of classification approaches in a pipeline. For example you may organise a classification system with the following sequence: handcrafted rules, induced rules for filtering messages into folders, and finally a SPAM filter approach. This could improve the overall performance by enabling each part to focus on what it does best.

This research also opens up the possibility of looking at the interplay and overlap between hand crafted and learnt rules. For example a user could scrutinise and modify a learnt rule and hence move it into the set of hand crafted rules. Moreover, the learning approaches could use the handed crafted rules to bias what is learnt. This could possibly producing a better set of rules.

8. Conclusion

In this paper we have described a composite rule learning algorithm that combines instance based approaches with rules. The clauses generated by this algorithm are compact and easy for users to understand. Moreover, we have show that this algorithm outperforms a wide range of other popular approaches to email classification.

We have also addressed some practical issues with incorporating automated email classification into email management software. In particular, we have discussed how hand-crafted and learnt rules can be used in tandem and demonstrated empirically the utility of our approach. In addition, we have tackled the problem of misclassification by outlining an interface design that minimises the cost of misclassification to users.

Acknowledgements

We would like to thank the people who contributed their email archives enabling us to conduct the empirical study. The authors would like to acknowledge the financial support of the Smart Internet Technology CRC. We also acknowledge and thank The University of Sydney for the financial support via the "Sesqui New Staff Support Scheme" grant which began this project in 2001.

References

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

Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C., & Stamatopoulos, P. (2000b). Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. *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.*

Bekkerman, R., McCallum, A., & Huang, G. (2004). *Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora* (Technical Report). Center for Intelligent Information Retrieval, University of Massachusetts, Amherst.

Bergadano, F., & Gunetti, D. (1994). An interactive system to learn functional logic programs. *Machine Learning.*

Boone, G. (1998). Concept features in re:agent, an intelligent email agent. *Second International Conference on Autonomous Agents.*

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

Cameron-Jones, M., & Quinlan, J. (1993). Avoiding pitfalls when learning recursive theories. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.* Morgan Kauffmann, San Mateo, CA.

Cameron-Jones, R., & Quinlan, J. (1994). Efficient top-down induction of logic programs. *SIGART Bulletin, 5*, 33–42.

Cohen, W. (1996). Learning rules that classify e-mail. *Papers from the AAAI Spring Symposium on Machine Learning in Information Access* (pp. 18–25).

Ducheneaut, N., & Bellotti, V. (2001). E-mail as habitat: an exploration of embedded personal information management. *interactions*, *8*, 30–38.

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

Klimt, B., & Yang, Y. (2004). The enron corpus: A new dataset for email classification research. *In the Proceedings of the European Conference on Machine Learning (ECML).*

Mackay, W. (1991). Triggers and barriers to customizing software. *CHI'91 Conference on Human Factors in Computing Systems* (pp. 153–160). New Orleans, Louisiana.

Mitchell, T. (1997). *Machine learning.* McGraw-Hill.

Mooney, R. J., & Califf, M. E. (1995). Induction of first-order decision lists : Results on learning the past tense of english verbs. *Journal ofArtificial Intelligence Research*, *3*, 1–24.

Pantel, P., & Lin, D. (1998). Spamcop: A spam classification & organization program. *Proceedings of AAAI-98 Workshop on Learning for Text Categorization* (pp. 95–98).

Pazzani, M. (2000). Representation of electronic mail filtering profiles: A user study. *Proc. ACM Conf. Intelligent User Interfaces.* ACM Press.

Provost, J. (1999). Naive-bayes vs. rule-learning in classification of email.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Rennie, J. (2000). ifile: An application of machine learning to e-mail filtering. *KDD-2000 Text Mining Workshop, Boston.*

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. *AAAI-98 Workshop on Learning for Text Categorization.*

Segal, R., & Kephart, M. (1999). Mailcat: An intelligent assistant for organizing e-mail. *Proceedings of the Third International Conference on Autonomou s Agents* (pp. 276–282). Seattle, WA.