IEMS - The Intelligent Email Sorter

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

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

Abstract

Classification of email is an important everyday task for a large and growing number of users. This paper describes the machine learning approaches underlying the i-ems (Intelligent-Electronic Mail Sorter) system. The system has two distinctive aspects: it offers a view of the inbox based on predicted classifications of messages; and it provides the user with details of the accuracy of the predictions and the processes underlying them.

We introduce a composite rule learner which classifies mail by combining an instance based approach with an approach which constructs a general explicit description. This was devised in order to achieve understandable, concise and effective classification rules. We report results of this learner as well as several others for data from for five users who apply very different ways of classifying their email. We discuss the implications of our results, in terms of the performance of the learning approaches, their sensitivity to concept drift and the ease with which the classification rules might be understood by users.

1. Introduction

This paper describes the i-ems (Intelligent-Electronic Mail Sorter) project which has the broad goal of improving our understanding of how to build systems which can assist users in managing email. In particular, we discuss work on automated support for classifying messages into appropriate folders. 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.

Users can be assisted in the task of classifying email if they make use of filtering rules available within many widely used mail interfaces such as Netscape and Microsoft Explorer. 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 applies 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 (Mackay, 1991) (Ducheneaut & Bellotti, 2001). In a recent study of user's management of email (Ducheneaut & Bellotti, 2001), the authors observe '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 has been motivated by the combination of the potential usefulness of mail management rules and the fact that so many users do not create them. This is designed to operate as an advising agent which suggests likely folders for storing a message where this might include a "Junk" folder for spam. 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.* Details of the interface may be found in Crawford et al. (2001).

This paper describes the exploration of several approaches for the automatic induction of email filtering rules. In Section 2, we describe related work and then, in Section 3 we introduce our instance based approach and Section 4 reports evaluations of that approach as well as several others. Section 5 concludes with a discussion of the results.

2. Previous 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 an *individual* user's classification of their own mail.

In particular, it is desirable that any learning algorithm should produce useful results quite quickly, with small amounts of data: the learning is for a single user's filing preferences and it is desirable that rules for automating this should be learnt from small numbers of examples.

Further, 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 also many other changes that affect classifications. For example, if the user's supervisor changes or other personnel at work change their roles, a learner will 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.

On the matter of the varying difficulty of learning an individual users' different mail classes, 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 Naive 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 Naive 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 the Keyword approach achieved precision of 95% and the Naive 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 the K-Nearest Neighbour) showing similar performance to a Naive Bayes classifier. Provost (1999) also evaluated Naive 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, Naive Bayes seems to have achieved precision around 95% in several studies. *Re:Agent* (Boone, 1998) also explored learning in a two-class case, this time 'work' versus 'other'. Boone used a hybrid approach with TF-IDF to learn useful features and then both neural networks and nearest neighbour approaches. This gave 98% accuracy on a dataset where the standard IR approach had 91% accuracy.

Although these quite high performance levels were achieved for two-category spam filtering, the more general categorisation task achieves weaker results. 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%. He observed that the rule based approach provided a more understandable description of the email filter. The *iFile* naive Bayes classifier (Rennie, 2000) was made available to several users to test on their own mail and this gave 89% accuracy. Brutlag (2000) assessed a Linear Support Vector Machine (SVM), reporting results from 70% to 90% correct and with the Unigram Language Model, 65% to 90%. They compared this against TF-IDF where they achieved 67% to 95%, depending on the store of email used.

This poorer result for general classification 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. In fact, the results summarised above seem very high for any real-

istic scenario where the user might have modest numbers of mail items in many of their mail folders.

The work described above suggests that automated classification should be able to operate usefully in helping users create classification mechanisms for their email. Unfortunately, without access to the test data in these studies, it is not clear how useful each would be in situations where users have modest numbers of mail items per folder.

The above work also indicates that some folder classifications will be far easier than others. 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. Then 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 these 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, they 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.

3. Using a richer hypothesis space

Instance based learning approaches, such as k-nearest neighbour, simply store the training data and use it directly to classify new examples. By contrast, learners such as c4.5 construct a hypothesis which is used independently of the training data to classify any new examples. Instance based approaches have the advantage of using the notion of locality and compare new examples with actual examples within the training set. They also have the advantage of being 'lazy' as they move the inductive step into the classification algorithm. Approaches that explicitly construct a hypothesis have the advantage of being able to generalise over the entire training set inducing a hypothesis that describes the concept independently from the training data. By combining these two approaches, advantages can be gained from both.

The learner we introduce uses clausal form to represent hypotheses. This is an ILP(Inductive Logic Programming) approach(Plotkin, 1971; Shapiro, 1981; Sammut, 1981; Muggleton, 1991), where the background knowledge is sophisticated and provides a strong bias for the learner. Also the search space is very limited. The hypothesis is constructed in a similar way to FOIL(Cameron-Jones & Quinlan, 1993) where clauses are constructed one by one in a top-down fashion. The approach is also like FLIP(Bergadano & Gunetti, 1994) where negative examples are not explicitly given: rather they are implicitly implied from the functional nature of the training set. Finally there is some similarity with FOIDL(Mooney & Califf, 1995) which permits intensional background knowledge, also the clauses generated are interpreted as first order decision lists where the first clause to explain a query is used. The email domain has also been previously investigated in an ILP setting, for example, (Shimazu & Furukawa, 1997). There are two distinctive aspects to the composite rule learner : firstly the search space is very tailored and restricted for learning in the email domain; and secondly the greedy covering approach of FOIL is not used but rather each clause is independently examined with respect to all the training data.

ILP enables the composite rule learner to combine an approach that explicitly constructs a hypothesis with instance based approaches. This permits the learner to shape and direct how the instance based approaches are applied. Note that, the hypothesis space is richer than simple propositional logic: literals that make up the clauses are relational, referring to elements within the training data. Consider the first clause in Table 1:

filter(M,F)⇐lastXsender_placements(F,M,5), F $\neq$ ml.

This clause states that a newly arrived message is predicted to belong in the same folder the use placed the last five messages from this sender. Also an exception is placed on this clause: it indicates that this clause should not be used for the "ml" folder. When the clause classifies new messages it makes use of messages currently in the folders.

Part of the appeal of this is that the clause captures much the same concept as a potentially large collection of clauses, each referring to a single sender. Also, it is more flexible in terms of concept drift. So, for example, if a user starts placing messages from a certain person into a different folder then this clause would stop activating. This would enable another clause to

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.

Table 1. Clauses induced by the 'Composite Rules' learner User 1 messages.

operate, potentially placing the message correctly.

The other instance based literals that are included in the scope of the learner are : contains_same(F, M, subject), contains_similar(F, M, subject, threshold, idf table), and contains_similar(F, M, body, threshold, idf table). The first of these "contains_same" is used to place messages into the same folder as a message with an identical subject.[1] The second and third of these "contains_similar" is used to place the message into the folder which has the most similar "subject"(or "body") provided it is greater than a ficed threshold.

chronologically sort T
$H := \{\}$
foreach $L \in$ four seed literals **do**
  $C :=$ "filter$(M, F) \Leftarrow L$"
  initialize $correct_f$ and $incorrect_f$ to map to 0
  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.

The approach for constructing the composite rule hypothesis is shown in Algorithm 1. This works as follows. The training messages are temporally ordered. Then each of the 4 key literals is considered in turn. An estimate is made regarding the number of messages the clause constructed from this literal would correctly filter. As this assessment is made the learner records how well the clause would perform on individual folders. Exceptions are added for individual folders if the clause classifies too many incorrectly in comparison to correct classification. Finally an assessment is made regarding the inclusion of the clause. This process is repeated for each of the possible clauses, which are then combined to form the final hypothesis.

Table 1 shows the set of clauses induced from data for User 1 in our evaluation experiments.

The beauty of the last two clauses shown in Table 1 is that the hypothesis gains the benefit of a similarity approach without being tied completely into it. This permits the exception conditions in combination with the other clause.

The hypothesis that is induced is in many respects independent of the data. This makes the approach more robust to the effects of concept drift. Also, as new aspects of the classification are introduced the hypothesis can deal with these aspects without relearning the entire concept. This approach has the potential to transfer to a number of other temporal domains.

Table 2 shows the hypothesis generated from another user, User 2 in our experiments. This user had 35 folders in comparison to the 7 of the first user. Also many of the folders ran orthogonal to the topic area of the messages. For example there was a folder "This_week" which related to communication that needed to be dealt within the week. As reflected in the empirical section, this makes the task of classification considerable more difficult. However, the "composite rules" approaches deals with this nicely by removing two of the clauses and adding a considerable number of exception condition(18 to the first clause and 25 to the second). Although this reduces the recall considerably, its precision is not completely decimated.

Note that, the algorithm could be tuned in a number of ways, these including: providing a non-uniform cost of misclassification, improving the heuristic for including the exception literals, and learning the threshold parameter in the seed literals.

4. Empirical Results

We now describe our experiments which compare the effectiveness of different learning approaches. We describe these approaches briefly, then the way that we

[1] This ignores a possible "Re:" at the beginning of the subject, also the comparison is done on a stemmed version of the subject.

filter(M,F)⇐contains_same(F,M,subject),
F ≠ M_ReplynDelete,
F ≠ This_week, ···, F ≠ Projects.
filter(M,F)⇐contains_similar(F,M,subject,0.6,idf),
F ≠ M_ReplynDelete,
F ≠ Junk, ···, F ≠ Projects.

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

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.

4.1 Learners Considered

The i-ems system uses an abstract class that permits a variety of learning approaches to be considered. Within this study we are considering five different approaches:[2]

- **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 (Cameron-Jones & Quinlan, 1994). 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 (Cohen, 1996) 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 (Segal & Kephart, 1999).
- **Naive Bayes** : This implementation of Naive Bayes 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 weighting their conditional probabilities during classification. We found that this increased accuracy. An email is classified as 'Unknown' by this approach if the probability it belongs to the best folder found does not exceed a threshold.
- **Composite Rules** : This is the approach described in the previous section.

[2]Within a previous study (Crawford et al., 2001) a simple decision tree learner also was considered.

4.2 Sliding Window 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 long 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.

It is extremely difficult to collect authentic data within this domain because of the private nature of some email communication. Our users were asked to volunteer classified email that they were comfortable with sending: this would clearly have altered their behaviour. However, we do not believe that this would have significantly perverted the findings of the this study.

We have conducted a preliminary evaluation of the five learning approaches using the above tester on a corpus from 5 users. The users have a variety of different approaches for classifying their messages. This makes some users' email easier to classify than others. For example, User 1 created folders purely on the basis of the area the messages related to. By contrast, User 4 created folders purely based on the action performed on the message(ie read, replied to, deleted, etc.) within the folder. These folders ran across topic areas. Hence, the learners for the 4th User performed poorly in comparison to the 1st User.

Table 3 show the number of messages and folders within each corpus. The testing window contains 30 messages and is moved in steps of 30 messages. The ac-

curacy is calculated as the percentage of the messages within the testing window that are correctly classified. We also calculate precision and the number of messages the learner labels "unknown". The results of these tests are shown in Tables 4 to 6. Frames indicate row maximums.

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

Table 3. Number of messages and folders for each of the users within the study.

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

Table 4. Accuracy of the different approaches. (S=Sender, K=Keyword, TI=TF-IDF, NB=Naive Bayes, CR=Composite Rules)

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

Table 5. Precision of the different approaches. (S=Sender, K=Keyword, TI=TF-IDF, NB=Naive Bayes, CR=Composite Rules)

We have also generated graphs of User 1 as the testing window is moved across the training data. These graphs are shown in Figures 1 to 5. [3]

The "TF-IDF" and "Keyword" learners have not classified any messages as '*unknown*'. Hence, when reporting on the performance over all the folders it is pointless distinguishing between precision and recall. The Sender, Naive Bayes, and Composite Rules approaches have made '*unknown*' classifications. Hence, we have also graphed the proportion of '*unknown*' classifications and the precision, where precision is defined

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

User	S	K	TI	NB	CR
1	47.5	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	38.8	41.0
4	44.9	5.1	0.0	54.4	96.9
5	31.7	31.6	0.0	68.2	29.1

Table 6. Percentage of unknown classification for the different approaches. (S=Sender, K=Keyword, TI=TF-IDF, NB=Naive Bayes, CR=Composite Rules)

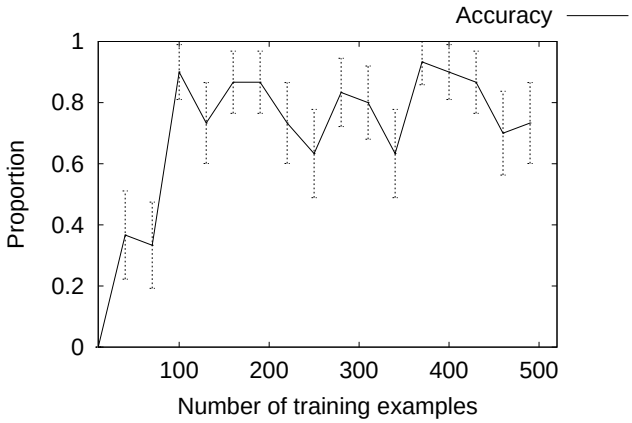

Figure 1. Graph of Accuracy for the Keyword Approach.

as the percentage of classified messages that are classified correctly.[4]

5. Discussion and Conclusion

With our goal of improving email management support, it is important to develop our understanding of how this task differs for different users. The experimental results reported above indicate considerable differences in the behaviours of the users in the study: there are corresponding differences in the experimental results. Notably, no one learner performs best for all users. Part of the explanation lies in the overall character of classifications. The best results were achieved for those users who classified mail on the basis of its topic. Poorer results were achieved where users classified mail according to when they needed to act on it or how they needed to respond to it.

Another striking observation is that the very simplest

[4]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, Naive Bayes 1 sec, and Composite Rules 6 sec.

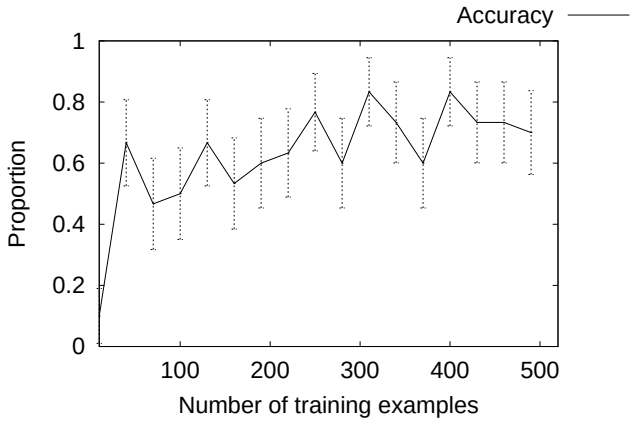

Figure 2. Graph of Accuracy for the TF-IDF Approach.

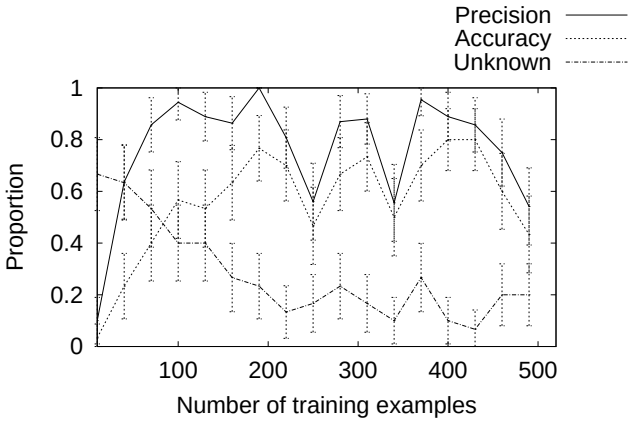

Figure 4. Graph of Precision, Accuracy and Unknown for the Naive Bayes Approach

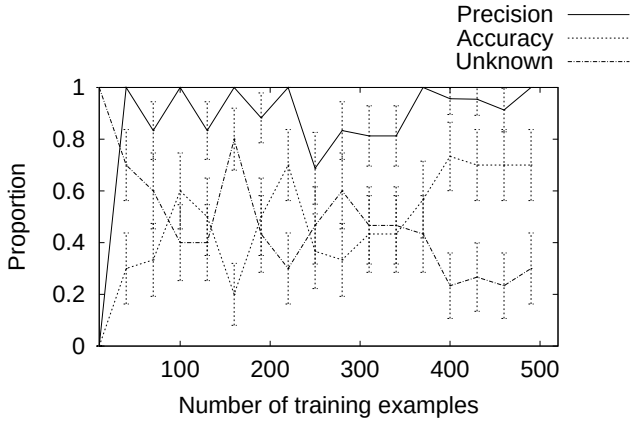

Figure 3. Graph of Precision, Accuracy and Unknown for the Sender Approach

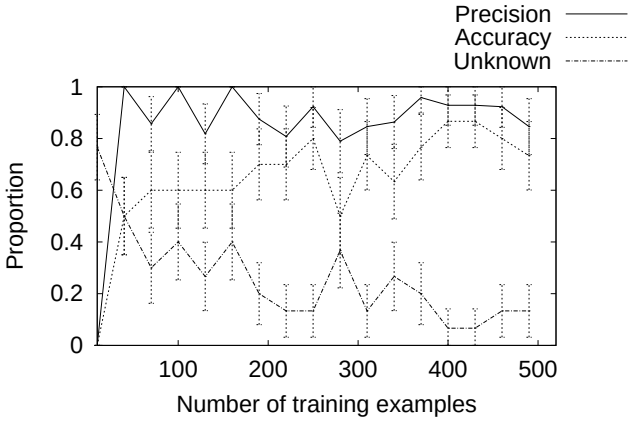

Figure 5. Graph of Precision, Accuracy and Unknown for the Composite Rules Approach

method, based purely on the sender of the email, performs remarkably well for some users. The Composite Rules learner performed quite well for some users. Notably, it had good accuracy for User 1 combined with a relatively low proportion of items classed as unknown. By contrast, on the data for User 4, the Composite Rules classifier assessed most of the items as unclassifiable and, on the very small proportion of items that it did classify, it achieved relatively good precision. User 2 had a somewhat intermediate set of results: the Composite Rules classifier had about half its items assessed as unclassifiable but this classifier had the best precision on those items that it did classify.

The data for the learning curves on the data for User 1 indicate that the TF-IDF approach learnt relatively slowly. By contrast, the Sender and Composite Rules classifiers learnt more quickly, requiring fewer messages. In this domain, it may be important to learn from less examples: this would enable support for the user after small amounts of data. At the same time, some mail folders tend to be quite large and good performance on those is important. So long term improvements may be useful.

The i-ems domain is email classification and filtering. This 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.

Acknowledgements

We would like to thank the people who contributed their email archives enabling us to conduct the empirical study. We also thank the reviewers for their valuable recommendations for improving this document.

References

Androutsopoulos, I., Koutsias, J., Chandrinos, K., & Spyropoulos, C. (2000a). An experimental compari-

son 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.*

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).

Crawford, E., Kay, J., & McCreath, E. (2001). Automatic induction of rules for e-mail classification. *In Proceedings of the Sixth Australiasian Document Computing Symposium, Coffs Harbour, Australia.*

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.

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.

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing, 8*, 295–318.

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.

Plotkin, G. (1971). *Automatic methods of inductive inference.* Doctoral dissertation, University of Edinburgh.

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

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.*

Sammut, C. (1981). *Learning concepts by performing experiments.* Doctoral dissertation, Department of Computer Science, University of New South Wales.

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.

Shapiro, E. (1981). *Inductive inference of theories from facts* (Technical Report 192). Computer Science Department, Yale University.

Shimazu, K., & Furukawa, K. (1997). Knowledge discovery in database by progol - design, implementation and its application to expert system building. *Proceedings of the 1997 ACM symposium on Applied Computing.*