Automatic Induction of Rules for e-mail Classification

Judy Kay and Eric McCreath

Basser Department of Computer Science
The University of Sydney
NSW 2006 Australia
{judy,ericm}@cs.usyd.edu.au

Abstract. Many users receive large amounts of email. Since a substantial part of that mail is kept for future reference, it is unsurprising that many mail tools allow users to create filtering rules that automatically do actions like saving mail in suitable folders. Unfortunately, most users do not make much use of this facility.
This paper describes the MUMILP project which will explore approaches to building a system to assist users in managing electronic mail. It will do this by learning rules for classifying email so that it can assist the user in filing messages.
We begin with a review of similar work and identify the core issues that the MUMILP project will explore: defining the terminology and ontology of a user's email domain; defining satisficing performance levels; building standard test sets for evaluating alternative tools that learn how to classify a user's email; machine learning approaches that give users good control over and understanding of the whole process of inferring email filtering rules; combining a user model holding the user's interests and goals with a machine learning tool which infers these rules; evaluation of ILP for learning the rules.

1 Introduction

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

This paper is concerned with the problems associated with management of email messages that the user wants to keep. Many email messages are simply read, then deleted. Many others are answered, then deleted. This leaves many mail items which are kept. One strategy for managing this email is to archive each piece into an appropriate folder. Choice of folder may depend on many factors including aspects such as the sender and nature of the email. For example, email from your manager may be filed into your *"manager"* folder. This task is non-trivial. For example, one study by Whittaker and Sidner [12] of 20 workers found

the average inbox had 2482 items. The average number of filed items was only 858. Within the study, some users left almost all their email in the inbox.

Many email programs[1] support filtering rules which can automate various mail management tasks: automated filing, deletion, replies. These rules can be expressed in terms of strings appearing in different parts of an email message. To handle an email item, the rules are evaluated in order and the first rule that applied to the item triggers the email client to move the message into the associated folder. The difficulty with rules is that the process of composing a rule is cognitively demanding and there is a real, potentially unacceptable risk of misfiling mail. Generally, users seem to avoid customising software [4].

The aim of MUMILP (Mail-management User Modelling with Inductive Logic Programming) is to investigate the automatic induction of email filtering rules. This might operate as an advising agent which suggests likely folders for storing a message. Alternatively, it might handle mail automatically, either at the stage of mail arriving or after the user has displayed it. At the same time, we want to provide an intuitive interface which enables users to scrutinise both the rules constructed for them and the *reasoning underlining the system's construction of these rules.*

2 Previous Work

A variety of approaches has been taken to address the problem of automating email classification. A list of approaches, in approximate chronological order, is given below:

- ***Learning Rules that Classify E-Mail*, by Cohen** [3]. Cohen uses the RIPPER learning algorithm that induces rules that spot keywords for classifying email. The paper compares this "keyword spotting" approach with a IR method based on TF-IDF weighting. Both approaches show similar accuracy. However, Cohen argues that keyword spotting is more useful as it induces an understandable description of the email filter.
- ***SpamCop*, by Pantel, and Lin** [7]. The objective of *SpamCop* is filtering junk email, hence its name. The system implements a Naive Bayes approach making use of both stemming and a stop-list. The stop-list is dynamically created by words that either occur less than 4 times in all the messages or by words that occur a similar proportion of time in both junk and legitimate messages. Note that this is different from classical Information retrieval stop lists which contain very common words. Pantel and Lin compared their system with Cohen's keyword-spotting approach *Ripper*.
- ***Re:Agent*, by Boone** [1]. Unlike other approaches *Re:Agent* is divided into two distinct stages. In the first stage, features are either learnt from collections of email messages[2] using a TF-IDF approach or they are con-

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

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

structed by users providing keywords. This permits the user to guide the agent without explicitly formulating rules. The second stage uses the features, constructed by the first stage, for learning actions. Boone investigates both neural networks and nearest neighbour approaches for this learning. Both these learning approaches are aided by the significant reduction in the dimensionality gained by first constructing the features.

- ***Bayesian Approach*, by Sahami, Dumais, Hecerman, and Horvitz** [**10**]. This system learns to filter junk email using a Bayesian Approach. As it is less desirable that legitimate messages are labelled as junk, than junk email is labelled as legitimate, a 99.9% threshold is used for classifying a message as junk. The investigation found that the incorporation of both phrases and domain specific features enhanced the systems performance.
- ***MailCat*, by Segal and Kephart** [**11**]. This system uses a TF-IDF approach which computes weighted vectors for each folder based on word frequencies, then a distance measure is used to estimate the similarity a new message has with each folder. When new messages were directly filtered into the most similar folder an error of 20% to 40% resulted. A later version took a slightly different approach: *MailCat* would give the user an option of the three most likely folders. This significantly improved performance. The user could then archive the email with a single mouse click on one of the three folder buttons.
- ***iFile*, by Rennie** [**9**]. Rennie uses a naive Bayes approach for text classification. The *iFile* works as a filter for the EXMH mail client. The system applies stemming and makes use of a stop-list. A number of *iFile* users were provided a program that performs a series of experiments on their own email folders; this helped address privacy issues relating to email. However, it also limits the ability of other researchers to perform a comparison with other approaches.
- ***Profile Usability*, by Pazzani** [**8**]. Pazzani reported on a usability study where users were asked to assess their preferences for different approaches for representing email filtering rules. The representations compared are: keyword spotting, perceptrons, and prototypes[3]. A set of messages are labelled as either "*discard*" or "*forward*". Then the subjects learnt how to classify these messages. This was achieved by providing unlabelled examples to the subject which they attempt to "guess". Feedback is provided and the subject's performance improves. The subjects are then asked to rate different representations of email filtering rules. This was in terms of their willingness to use these rules for performing the same decision they had just learnt. The results showed that subjects preferred the prototype representation. Also

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

the study found that word pairs, as opposed to individual words, were preferred by the subjects. This is a significant study as it considers the user's preferences in representing mail filtering rules.

	Folders v 2 Classes	Key Word Spotting	TF-IDF	Naive Bayes	Percep-tron	Proto-type	Feature
RIPPER[3]	Folders	87-94%	85-94%				
SpamCop[7]	2 Classes	86%		94%			
Re:Agent[1]	Folders		72%				87%
Sahami, et al [10]	2 Classes			95%			
Pazzani [8]	2 Classes	X			X	X	
ifile[9]	Folders			89%			
MailCat[11]	Folders		60-80%				

Table 1. Accuracy results reported by different approaches. Note that, Pazzani[8] reported on a usability study that considered types of profiles people would be willing to use. Hence, no accuracy results are provided.

Table 1 provides a summary and comparison of these systems: their approaches and reported accuracy results. These results are not comparable between systems as each uses different training/test examples. Most of the studies consider a variety of parameters for the different systems; the table only shows the best result obtained from the largest data-set.

3 Outstanding Problems

Although it may not be clear from the description above, this domain seems typical of many where machine learning is intended for user modelling. In particular, it is desirable that the learning 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.

3.1 Terminology and ontology

We can see this as one of the elements common to several of systems described. Essentially, this is a first step in learning email filtering rules: it involves finding the terms to use within mail documents. This gives as basic terminology for the user model for an individual.

It should include the terms with the greatest potential to be useful for the filtering tasks. Just as in classic Information Retrieval systems, this should exclude terms on a stop list of common words. Since these occur in most documents, they have poor predictive power for classification. They also increase the time complexity of the subsequent machine learning.

In addition, some of the systems create a stop list with the terms that occur very infrequently. Zipf's Law means that there will be very large numbers of terms that occur very infrequently. One might argue that the many terms which appear in only one document might be discarded. The reasoning is that they are useless for identifying document groupings since they occur in only one document. Further, if there are large numbers of them, they may cause a dramatic increase in the time and space complexity of the machine learning algorithms. On the other hand, we should note that this approach is not taken in information retrieval systems: if some of these rare words appear in a new document, it may be a good indicator of the classification for that document. This is the basis of the TFIDF approach. At this stage, this difference deserves further exploration.

The first vocabulary comes from terms in the actual email items. The email classifier will try to identify these. A second vocabulary is defined by the folder names. Both these sets of terms can be included in a user model. In addition, a rich user model might include elements in a third vocabulary which might assist in the mapping from the first set of terms to the second.

It seems likely that one could identify relationships between elements in these vocabularies. These could be useful both for the machine learning and for the construction of the user model ontology. One very simple instance of this has been explored in some of the systems where word pairs are used. We could view this as defining relationships between the terms.

Perhaps more interesting is the definition of an ontology on the classes to be learnt. These will probably constitute meaningful categories for the user. For example, in a hierarchical system of mail folders, we can define a corresponding hierarchy of relationships between the classes. This is an area that deserves exploration both as an aid to the machine learning and as a structuring mechanism for defining an ontology on the user model vocabulary.

3.2 Satisficing performance

In systems like *MailCat*, it is reported that error rates of 20-40% were judged to be inadequate, while rates over 90% were satisfactory. It is clear that this is a domain where it is important to achieve a satisfactory level of performance: beyond that, small reductions in error rate may not matter much. Unfortunately, we have no sense of just what level of performance is satisfactory.

3.3 Standard data sets

In reviewing the work described in the last section, it is difficult to compare the performance of the different systems. One of the problems is that each was evaluated on different data. To improve our understanding of the relative performance of different machine learning approaches in this domain, it seems desirable to be able to compare their performance on common, standard data sets.

3.4 User control and understanding

In this domain, the user model that we want to learn will, like most user models, contain personal information. The task managed by filtering rules may also be quite important to the user. It seems desirable to explore machine learning approaches that give users control over, and understanding of, the whole process of inferring email filtering rules.

3.5 Broad user model as aid to learning

Although the systems we have described evaluate various approaches to machine learning, none tries to augment the learner with a user model holding the user's interests and goals.

3.6 Evaluation of ILP

The usefulness of ILP for classifying email messages is still an open questions. We propose to explore its effectiveness, compare this with the effectiveness of other approaches, such as those described in Section 2 and then we will evaluate user responses to the set of rules and explanations for them.

4 Proposed Approach

The MUMILP project will tackle the issues just described. This section describes the two main tasks we are undertaking.

4.1 Public Data Set

It would be useful to have a publicly available data set. This would provide a better comparison of the performance of different approaches. However, there are significant ethical problems in making such data publicly available. Often an email message is sensitive. Hence, it would be unlikely that either the sender or receiver would consent to such information being made public. Also given the large number messages/senders in a typical email archive it would be a logistically enormous task to obtain such consent. Hence, we argue that a synthetic data set would both address the ethical concerns and provide a common data set for comparing approaches. This data set would be made available to the public so that other researchers can compare the performance of their approaches.

4.2 An ILP Approach

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. This is a similar problem with a Naive Bayes approach.

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.

We intend to develop a system that uses a richer language for describing profiling rules. This would still provide a profile that is scrutable. Also it would address some of the "brittleness" issues that flow from the dimensionality of the problem. The approach will use a restricted first order language for representing profiles and it will make use of a variety of ILP techniques. We plan that the profile would closely match the way that real users model and filter messages.

ILP(Inductive Logic Programming) involves systems that learn a restricted form of first order logic. A variety of systems have been implemented and studied These include: Foil[2], Progol[6], and Lime[5]. There is a number of issues that arise when applying ILP to such a domain, including: handling the large number of training examples, incremental learning, problems due to concept drift, and how the learner may be 'biased' to improve the predictive accuracy. Given the enormous amount of information in a person's email archive it must be pre-processed before it is provided to the ILP system. This will form a bias on the learner and will affect the overall performance of the learner.

Rather than explicitly define the way profiles will be represented we illustrate our intended approach with an example. Suppose a user has a variety of folders including "friends" and "accounting". Messages that are sent from personal friends are directed into the "friends" folder. This is primarily dependent on the sender's identity. Messages relating to accounting and finance are placed in the "accounting" folder. This is dependent on either the content of the message, where a similarity measure could be used, or if the message is from 'Bill' the organisation's accountant, then it could be directly moved into the "accounting" folder. We may express rules to filter these message in first order form as follows:

$$\text{filter}(M, \text{accounting}) \leftarrow \text{sender\_identity}(M, \text{“Bill”}), !.$$
$$\text{filter}(M, \text{friends}) \leftarrow \text{sender\_identity}(M, X), \text{in}(X, [\text{“Jack”}, \text{“Jill”}]), !.$$
$$\text{filter}(M, \text{accounting}) \leftarrow \text{messages}(X, \text{accounting}), \text{most\_similar}(X, M), !.$$

There are three important factors to take note of in this representation:

- Rule based and TF-IDF approaches may be combined by using literals such as "similar" or "most_similar".
- It is possible to generate higher level features such as "sender_identity". This is intended to address some of the "brittleness" and dimensionality problems that arise.
- Also rules may be ordered providing a mechanism to give some rules priority over other rules.

5 Discussion and Conclusions

Within the ILP framework, the MUMILP project will explore the outstanding problems we described earlier. We propose to explore the impact of an initial user model with details of user interests. This will provide one dimension of the background knowledge available to the learner.

To properly study the accuracy of the email agent it is critical that the email agent is incorporated into a widely used mail client. This part of the project is also critical for studying usability and scrutability of the approach. It is also important for the MUMILP goal of defining the levels of performance necessary for user satisfaction.

The MUMILP domain is email classification and filtering. This domain has many of the elements that are common to emerging areas for personalisation and it should improve our understanding of the potential for ILP in scrutable user modelling.

References

1. G. Boone. Concept features in re:agent, an intelligent email agent. In *Second International Conference on Autonomous Agents*, may 1998.
2. R.M. Cameron-Jones and J.R. Quinlan. Efficient top-down induction of logic programs. *SIGART Bulletin*, 5(1):33–42, 1994.
3. W. Cohen. Learning rules that classify e-mail. In *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, pages 18–25, 1996.
4. W.E. Mackay. Triggers and barriers to customizing software. In *CHI'91 Conference on Human Factors in Computing Systems*, pages 153–160, New Orleans, Louisiana, 1991.
5. E. McCreath and A. Sharma. Lime: A system for learning relations. In *The 9th International Workshop on Algorithmic Learning Theory*. Springer-Verlag, October 1998.
6. S. Muggleton. Inverting implication. In S. Muggleton and K. Furukawa, editors, *Proc. of the Second Int. Workshop on Inductive Logic Programming (ILP92)*, 1992.
7. P. Pantel and D. Lin. Spamcop: A spam classification & organization program. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, pages 95–98, 1998.
8. M. Pazzani. Representation of electronic mail filtering profiles: A user study. In *Proc. ACM Conf. Intelligent User Interfaces*. ACM Press, 2000.
9. J. Rennie. ifile: An application of machine learning to e-mail filtering. In *KDD-2000 Text Mining Workshop, Boston*, 2000.
10. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
11. R. Segal and M. Kephart. Mailcat: An intelligent assistant for organizing e-mail. In *Proceedings of the Third International Conference on Autonomou s Agents*, pages 276–282, Seattle, WA, 1999.
12. Steve Whittaker and Candace L. Sidner. Email overload: Exploring personal information management of email. In *CHI*, pages 276–283, 1996.