

# Iems: helping users manage email.

Eric McCreath<sup>1</sup> and Judy Kay<sup>2</sup>

<sup>1</sup> Department of Computer Science,  
The Australian National University,  
ACT 0200 Australia  
[ericm@cs.anu.edu.au](mailto:ericm@cs.anu.edu.au)

<sup>2</sup> School of Information Technologies,  
The University of Sydney,  
NSW 2006 Australia  
[judy@it.usyd.edu.au](mailto:judy@it.usyd.edu.au)

**Abstract.** This paper reports our work to build an email interface which can learn how to predict a user's email classifications at the same time as ensuring user control over the process. We report our exploration to answer the question: does the classifier work well enough to be effective? There has been considerable work to automate classification of email. Yet, it does not give a good sense of how well we are able to model user's classification of email. This paper reports the results of our own evaluations, including a stark observation that evaluation of this class of adaptive system needs to take account of the fact that the user can be expected to adapt to the system. This is important for the long term evaluation of such systems since we may find that this effect means that our systems may be performing better than classic evaluations might suggest.

## 1 Introduction

As people need to cope with increasing amounts of email, there is a corresponding growth in the need to support users in organising mail and attending to it more effectively. 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.

A good foundation for this work comes from the studies of the ways that people manage email within current email clients. Mail folders would seem to offer a useful organisational structure for email. Yet these facilities appear to be of limited value for many users. For example, in one study of 20 workers [1], the average inbox had 2482 items, with an average of 858 filed. Some users kept most of their email in the inbox. When users do classify email, Ducheneaut and Bellotti [2] report that this is commonly based on quite simple criteria: sender name or organisation, project or personal interests.

Most mail clients allow users to write rules which will automatically sort email into folders. This has limited value for many users. Generally, users avoid

customizing software [3, 2] and many users avoid such rules, as they are too difficult to create.

There has been considerable work in the use of machine learning to automatically classify email. An interesting subproblem is the identification of junk mail. This has been explored with a variety of machine learning techniques, for example, Naive Bayes [4, 5, 6], Bayesian approaches [7], genetic approaches [8], keyword approaches [5], a memory-based approach similar to K-Nearest Neighbour [9], the RIPPER algorithm [6]. These typically report precision and accuracy measures around 95% or better and recall values of 70-80%.

Broader classification has, as one would expect, achieved less success. Cohen [10] used RIPPER, achieving 87%- 94% accuracy and TF-IDF (85%-94%). A Naive Bayes classifier [11] gave 89% accuracy. Brutlag and Meek [12] compared a Linear Support Vector Machine (70% to 90% correct), the Unigram Language Model (65% to 90%) and TF-IDF (67% to 95%) with the range due to different email sets. As a body, these results are encouraging. In spite of the non-comparability of these studies, they give an overall indication that a variety of approaches can achieve average accuracy of 70-90% and a lesser level of recall.

Central to our i-ems project is the ability to measure the effectiveness of approaches that are currently available as well as those we create. The email classification domain has some special evaluation challenges. Notably, there is the issue of user consistency. This has been observed, for example, in the case of human classifiers of the Reuters data [13]. The work listed above tends to report accuracy, precision and recall. All appears to have been tested by taking a set of preclassified email, testing on part of it and then evaluating on the remainder. If some of that was actually misclassified, it is harder to assess the actual effectiveness of an approach.

A different and interesting approach to both the classification of email and evaluation of effectiveness was taken in *MailCat* [14]. This interface provides three buttons showing the three best predicted mail folders for the current message. This approach increased the chance of offering a correct folder. It is interesting in that this design of the interface ensures a modest cost for incorrect classifications. Moreover, they have reported both the accuracy and user's affective assessment: their earlier single button design, with 20% to 40% error rates was considered unacceptable while users were satisfied with the three-button interface. A repeat study [15] supported the claim of user satisfaction. Our approach takes a similar philosophy, with a focus on the interface and the role of user involvement in evaluation processes.

In Section 2, we describe the i-ems interface. Then, in Section 3, we describe experiments which evaluate the relative power of hand crafted mail classification rules, rules learnt automatically and a combination of these. We also report an unexpected observation, the effect of the system's recommended classification affecting the user's classification of messages. In the final section, we discuss the findings, especially the matter of performance metrics for such systems.

## 2 IEMS - The Electronic Mail Sorter

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 messages in the folder that is currently selected. In the figure, this is the inbox. This shows each message under its predicted folder category or under the category *Unknown* if no prediction has been made. For example, the messages sent from Judy Kay have been predicted to belong in the *crc* folder. Each appears as a single line with the sender and the subject. The user has currently selected one of these messages from Judy Kay. This message is displayed in the lower right-hand window.

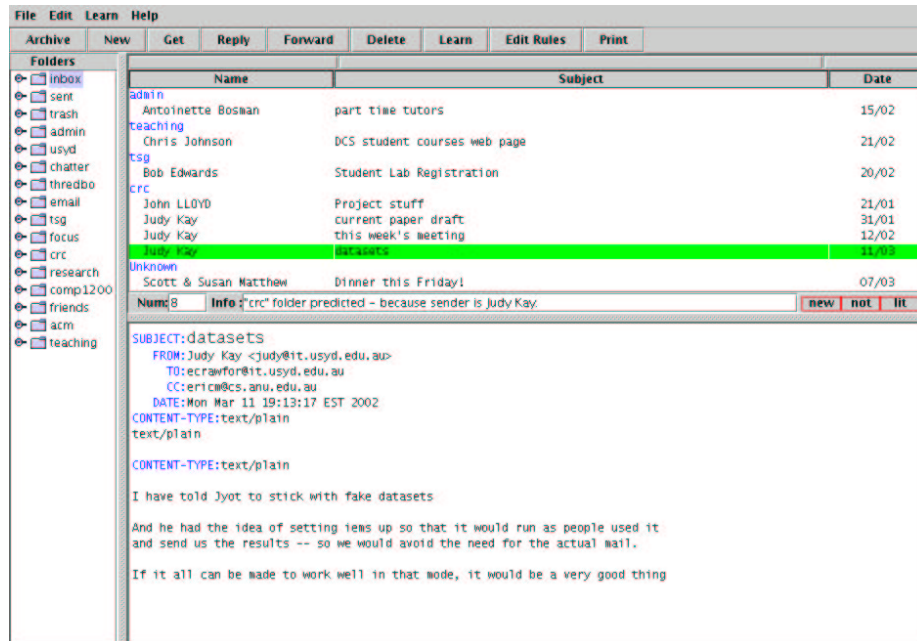


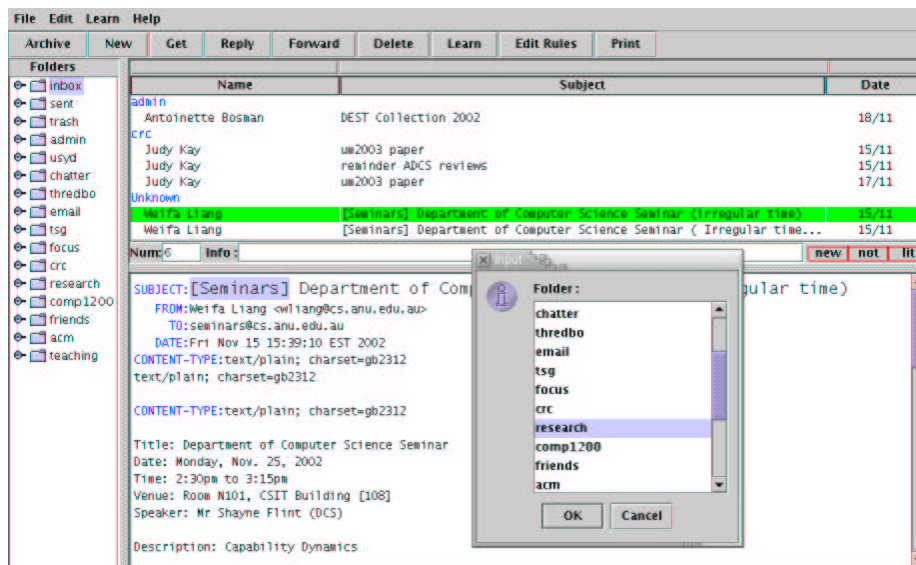
Fig. 1. A screen shot of the i-ems interface.

Once a user has read a message, there are two possible courses of action. If they are happy with the classification, they can simply click on 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 recommendation. In this case, the user can simply drag and drop the message into the desired folder. This is much the same amount of effort required for a user to archive messages in a standard email manager.

At times, a user may wish to direct and assist these recommendations. This is especially the case when the user knows that particular class of messages will be filed into a particular folder. For example a user may know that all messages with “[seminar]” in the subject will be filed into the “research” folder. The user may simply add such a rule by doing the following: highlighting the “[seminar]” keyword in the subject of a message, clicking on the “new” button to the right of the information text field, and selecting the “research” folder in the dialog box that appears. This is shown in Figure 2. In a similar way, the user may refine existing rules by using the “not” and “lit” button. These respectively add negated and ordinary literals to a rule. As hand crafted rules tend to be more precise, they are given priority in classifying new messages over the learnt rules. Hence, the learnt rules are only used when the hand crafted rules do not make any classification.

Another important aspect of our approach is that i-ems allows the user to scrutinise the classifier. The interface shows the user the reason 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 given the task of generating these explanations. If the classification resulted from a hand crafted rule then this rule is simply appears here.



**Fig. 2.** A screen shot of adding a new rule with the i-ems interface.

The user is thus able to both scrutinize why the recommendations are made and refine, augment, and control these recommendations. This gives the user a sense of control over their email manager while still aiding them.

### 3 Empirical Results

We now describe our experiments which show two characteristics of the email system. First, we show how the combination of hand crafted and learnt rules can be more effective than either approach working alone. Second, we see how the use of a particular learning approach within the email manager influences the way the user classifies messages. However, before we look at these results we describe: the learners, the testing, and the dataset used.

The i-ems system uses an abstract class that allows different learning approaches to be considered. In previous work, [16, 17] we evaluated a number of learners. However, within this study just two learners are considered:

- **Sender** : This learns rules to predict the folder for a new message based solely on the sender. For each sender, a rule is created to filter new messages into the folder that the sender most commonly goes into. Mail from new senders is classified as “unknown”. We realised this was a very simplistic and limited approach. However, it has shown to be effective [17].
- **Keyword** : The keyword approach induces a set of clauses in a similar way to Quinlan and Cameron-Jones’ FOIL [18]. The literals considered are whether a particular word is contained in one of the : sender, to, subject, or body fields. This operates somewhat like Cohen’s Ripper [10] in that it induces rules for the smallest folder first, progressing to larger ones.

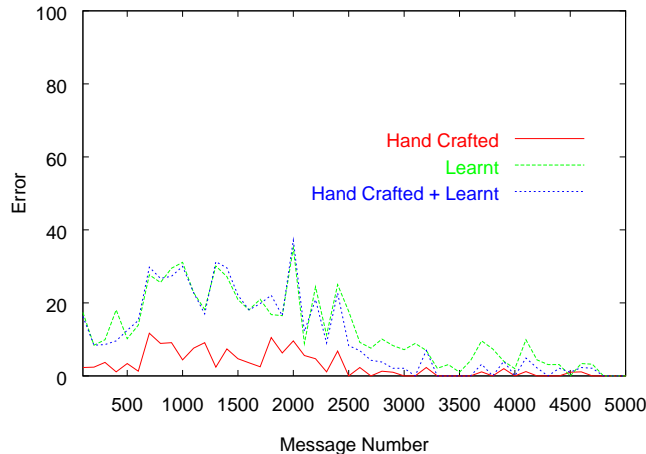
A cross validation testing approach would be inappropriate given the temporal nature of email. A sequence of email messages will often contain threads of discussions relating to particular topics. If cross validation were used, the training data may always contain messages from these threads, hence simplifying induction. When an email manager is used it will have an initial sequence of messages and must predict the classification of future messages. We reflect this in our evaluation approach. A sequence of messages is presented to the learner chronologically ordered and then we test on the subsequent collection of messages. This process is repeated giving a “sliding window” tester where the testing window is moved across the examples and all the messages prior to the window are provided for training. This testing approach has the merit of reflecting the way a learner within a mail client would operate.

We wanted to test on a substantial corpus of mail. We also wanted a complete set. This meant that all mail should be kept and classified. In the case of mail that the user would have normally deleted, we wanted that mail classified into a folder. Other than this, we wanted the user to choose the folder names and associated concepts as they wished. So, for example, in some of our experiments, users have created a “deleted” folder for mail they would have deleted. Others made folders called “readndelete” to indicate what that person did with those messages.

Our previous work involved five users who were willing to classify mail as required [16, 17]. For this experiment, we collected a much larger data set over a longer period. This was for a single user, who had participated in the earlier work.

The large corpus of messages contains 5100 messages spanning approximately 3 months of email activity. The user has 21 folders and 70 hand crafted rules giving the system a nontrivial real world test. By conducting the empirical study only on one user, we limit the generality of any conclusions we can make. However, as seen in previous studies[16, user 1][2], the way the user archives messages is characteristic of at least a sub-group of users. Hence, we argue that our results are valid for at least a subgroup of users.

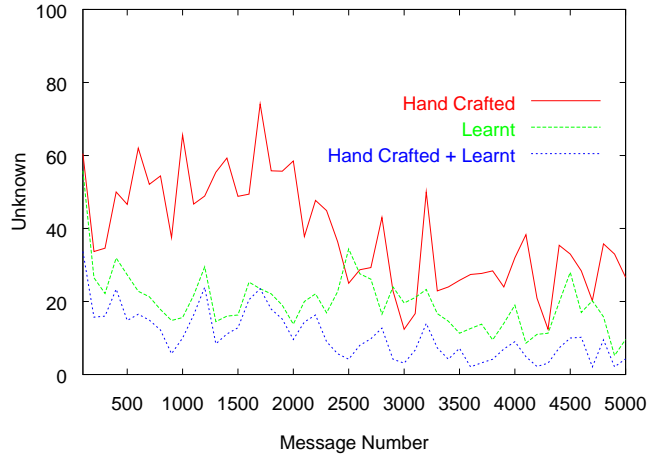
The testing window contains 100 messages and is moved in steps of 100 messages. The “error” is calculated as the percentage of the messages within the testing window that are incorrectly classified. This shows the percentage, over all the messages, where the rules make a definite classification and this classification is incorrect. We also calculate “precision” which is the percentage of messages correctly labeled that were not labeled “unknown”. Also the percentage of messages the learner labels “unknown” is given.<sup>3</sup> The results of these tests are shown in Figures 3 to 5.



**Fig. 3.** Sliding window test results showing error.

Figure 3 shows the error rate for the three approaches, with the hand-crafted rules performing best. Figure 5 shows that the hand crafted rules are consistently more precise than the learnt rules. However, as seen in Figure 4, the hand crafted rules classify more messages as “unknown”. The combination of the learnt and handcrafted rule always has a lower level of “unknown” messages, this is because the learnt rules attempt to guess the classification of the messages the hand

<sup>3</sup> “Error”, “precision”, and “unknown” are defined as follows:  $\mathbf{error} = 100 \times \frac{i}{u+c+i}$ ,  $\mathbf{precision} = 100 \times \frac{c}{c+i}$ ,  $\mathbf{unknown} = 100 \times \frac{u}{u+c+i}$ , where  $i$  is the number of message incorrectly classified,  $u$  is the number of messages classified as “unknown”, and  $c$  is the number of messages correctly classified.



**Fig. 4.** Sliding window results showing unknown.

crafted rules considered “unknown”. So by combining the two approaches the number of messages classified as “unknown” is reduced while still maintaining a similar precision and overall error. This result is also reflected in the Table 1 which shows the average performance on the first half of the corpus, split equally between training and test set. If a user’s primary criteria is reducing the error rate, then hand crafted rules are best. However, the hand crafted rules alone are less useful as they make considerably less predications, so many of the messages that come into the users inbox would be labeled “unknown”. We argue that by combining the approaches, with a small cost in error rate, the number of unknown messages can be considerably reduced. This would be more useful for many users.

One striking and interesting aspect of these graphs is the sharp improvement in performance of the system after about 2500 messages. This is because the first 2500 messages were collect and archived simply using the drag and drop facility, whereas, the archiving of the second 2500 messages was aided by the sorting of the inbox using a combination of the hand crafted and learnt rules (using the sender approach). Clearly the use of such a system influences the way in which a user will archive messages. Hence, the utility of measuring the performance of a

Rules used	Error	Unknown	Precision
Hand crafted	7.1%	52.4%	85.0%
Learnt only	20.2%	29.1%	71.5%
Hand crafted + learnt	22.5%	20.0%	71.9%

**Table 1.** Comparison of combining learnt and hand crafted rules using the Sender learner training on messages 0–1249 and testing on messages 1250–2499.

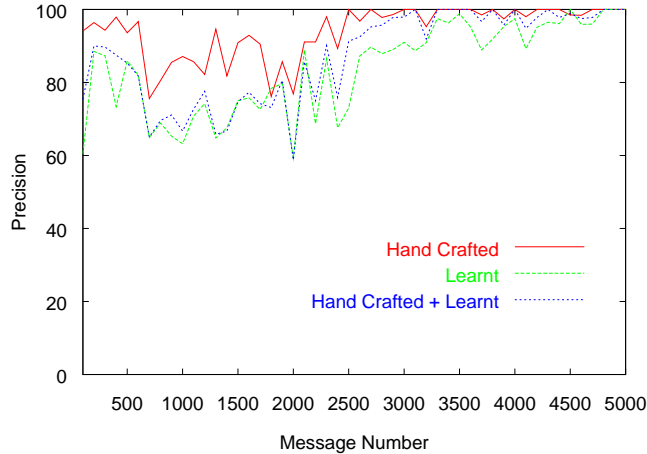


Fig. 5. Sliding window results showing precision.

particular system in isolation is questionable, as recommendations provided by the system will influence the archiving of messages. This is particularly the case in email sorting, as often folders will have blurred and overlapping intentions. So when a system recommends a particular folder it may not be the folder the user would have used without such a recommendation. However, it may be 'close enough', in which case the user will archive it directly into the recommended folder rather than dragging and dropping it into another folder. It may even be that the user would be inconsistent in their classification and that i-ems is helping them to maintain consistency. This is clearly reflect in Table 2 where we use the 'sender' approach (combined with the hand crafted rules) and the system trains on the first 2500 messages and we test on the next 2500 message the overall error is reduced to 2.7% and the unknown percentage is only 9.3%.

Note that, as the 'sender' approach was used in aiding the sorting of messages, the sorting of messages is particularly biased toward this learner. When we train and test the systems prior to the user being aided by the sorting of messages within the inbox the 'sender' and 'keyword' approaches both produce very similar results. However, when we expand the test into the region where the user is aided

Training/Testing messages	Learner Used	Error	Unknown	Precision
Training 0-1249/ Testing 1250-2499	Sender	22.5%	20.0%	71.9%
	Keyword	22.1%	22.8%	71.3%
Training 0-2499/ Testing 2500-4999	Sender	2.7%	9.3%	97.0%
	Keyword	16.2%	5.4%	82.8%

Table 2. A comparison of the sender and keyword approaches when they are incorporated into the system.

by the sorting of messages within the inbox the difference between the sender and the keyword approach is stark. These results are shown in Table 2.

## 4 Discussion and Conclusion

The i-ems project explores a range of approaches to support users in managing email information overload. 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 tools that users can readily understand and control.

The design of the i-ems interface was driven by the need to operate with flawed classifiers. It ensures a modest cost in the case of misclassification. It also has several levels at which it ensures a real sense of user control: the user can create classification rules; the user can see the reason for the classification of each mail item; and the user can refine and augment classification rules, including those learnt by the system. We have achieved important additional functionality. When i-ems organises the inbox according to the categories, it effectively groups related mail. This means that the user can maintain focus on the context represented by the items of one folder, while the mail is still in the inbox.

This paper reported an extensive experiment with over 5000 email items. It has reported evaluations with two classifiers and the potential for combining automated machine learning and user constructed rules. The generality of our claims is limited by the fact that this work was restricted to a study of one user's mail management. We note, however, that our previous work [16, 17] and broader studies of users [2] suggests that this has considerable generality.

Our evaluation strongly suggests that when the user was presented with an inbox, organised into classes predicted by i-ems, they were influenced by this classification. This has important implications for the evaluation of such systems. We believe that an important direction for future evaluations of such systems should include field tests similar to that we have reported. This approach takes account of the known problem of inconsistent classification by the user [13]. Indeed, in personalised document classification, this is apt to be a significant factor. It also measures success in terms of the individual judgement of the user: an item need only be regarded as misclassified if the user takes the trouble to correct it. Like the *MailCat* [14, 15] evaluations, it essentially has an affective evaluation built in. While it may seem more rigorous to train and test classifiers on preclassified email, we believe that a field test such as we have reported should also be used to give a better answer to our core question: how effective is this classifier and interface for this user?

**ACKNOWLEDGMENTS** We would like to thank the Smart Internet Technology - Corporate Research Center for their support of this research. We thank the reviewers for their valuable recommendations. Also thanks to Daren Ler for help in proof reading this document.

## References

- [1] Whittaker, S., Sidner, C.L.: Email overload: Exploring personal information management of email. In: CHI. (1996) 276–283
- [2] Ducheneaut, N., Bellotti, V.: E-mail as habitat: an exploration of embedded personal information management. *Interactions* **8** (2001) 30–38
- [3] Mackay, W.: Triggers and barriers to customizing software. In: CHI'91 Conference on Human Factors in Computing Systems, New Orleans, Louisiana (1991) 153–160
- [4] Pantel, P., Lin, D.: Spamcop: A spam classification & organization program. In: Proceedings of AAAI-98 Workshop on Learning for Text Categorization. (1998) 95–98
- [5] Androutsopoulos, I., Koutsias, J., Chandrinou, K., Spyropoulos, C.: An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. (2000) 160–167
- [6] Provost, J.: Naive-bayes vs. rule-learning in classification of email. Technical Report AI-TR-99-284, University of Texas at Austin, AI Lab (1999)
- [7] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk e-mail. In: AAAI-98 Workshop on Learning for Text Categorization. (1998)
- [8] Katirai, H.: Filtering junk e-mail: A performance comparison between genetic programming & naive bayes (1999)
- [9] Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C., Stamatopoulos, P.: Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In: Proceedings of the Machine Learning and Textual Information Access Workshop of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD. (2000)
- [10] Cohen, W.: Learning rules that classify e-mail. In: Papers from the AAAI Spring Symposium on Machine Learning in Information Access. (1996) 18–25
- [11] Rennie, J.: ifile: An application of machine learning to e-mail filtering. In: KDD-2000 Text Mining Workshop, Boston. (2000)
- [12] Brutlag, J., Meek, C.: Challenges of the email domain for text classification. In: Seventeenth International Conference on Machine Learning. (2000)
- [13] Apte, C., Damerau, F., Weiss, S.M.: Automated learning of decision rules for text categorization. *Information Systems* **12** (1994) 233–251
- [14] Segal, R., Kephart, M.: Mailcat: An intelligent assistant for organizing e-mail. In: Proceedings of the Third International Conference on Autonomous Agents, Seattle, WA (1999) 276–282
- [15] Ruvini, J.D., Gabriel, J.M.: Do users tolerate errors from their assistant?: experiments with an e-mail classifier. In: Proceedings of the 7th international conference on Intelligent user interfaces, ACM Press (2002) 216–217
- [16] Crawford, E., Kay, J., McCreath, E.: Automatic induction of rules for e-mail classification. In: In Proceedings of the Sixth Australasian Document Computing Symposium, Coffs Harbour, Australia. (2001)
- [17] Crawford, E., Kay, J., McCreath, E.: Iems - the intelligent email sorter. In: In Proceedings of the Nineteenth International Conference on Machine Learning, 2002, Sydney, Australia. (2002)
- [18] Cameron-Jones, R., Quinlan, J.: Efficient top-down induction of logic programs. *SIGART Bulletin* **5** (1994) 33–42