# EMMA: An E-Mail Management Assistant

Van Ho, Wayne Wobcke and Paul Compton
*School of Computer Science and Engineering*
*University of New South Wales*
*Sydney NSW 2052, Australia*
{*vanho*|*wobcke*|*compton*}*@cse.unsw.edu.au*

## Abstract

*In this paper, we describe EMMA (*E-Mail Management Assistant*), an e-mail system that addresses the process of e-mail management, from initially sorting messages into virtual folders, to prioritizing, reading, replying (automatically or semi-automatically), archiving and deleting mail items. EMMA attains a high degree of accuracy on e-mail classification by using a rule-based approach known as Ripple Down Rules (RDR) as the basis of rule construction. In contrast to traditional rule-based systems, RDR systems provide extensive help to the user in defining rules and maintaining the consistency of a rule base, making EMMA easy to use. We discuss the results of evaluating the usability of EMMA on a trial of independent users.*

## 1. Introduction

As e-mail becomes a more prevalent form of communication, dealing with e-mail is becoming more and more costly and time consuming. There is a real need for software assistants that can improve the management of personal and organizational e-mail. But despite much research into the development of e-mail assistants, current e-mail clients and research prototypes provide only limited help in addressing this problem of communication overload.

Maes [5] identifies competence and trust as two main criteria for the effectiveness of personal assistants: *competence* refers to the acquisition of knowledge by the assistant and the use of this knowledge to aid the user; *trust* refers to the user's willingness to delegate tasks to the assistant. Of course, this assumes that the assistant is "competent" to the extent that the user model it develops is correct and able to be employed to help the user. In this more everyday sense, competence refers to the correctness of the assistant's behaviour. To these, we can add *usability*; the ease of interaction between the user and the assistant, including the ease with which knowledge is acquired by the assistant.

In the case of e-mail, user group studies, e.g. Singh and Ryan [8] have endorsed Maes' view by showing that users' lack of trust in their systems is a major barrier to the use of e-mail in organizations (and this would apply even more to the use of e-mail assistants). We believe that the main reason that existing prototype e-mail assistants are not in widespread use is that such systems (typically based on machine learning techniques such as Bayesian classification) do not provide competence (in the sense of correctness) and hence do not engender the trust of their users.

In this paper, we focus on e-mail management as a process. Considering a single e-mail message, the message first arrives in a user's Inbox and is perhaps sorted and/or prioritized for display to the user before being read. The message may be left in the Inbox for some time before the user replies to it, then it may either be archived and/or eventually deleted. Archived messages may be culled at a later date. We believe that another reason existing e-mail assistant prototypes are of limited use is that they typically address only one of these aspects of the e-mail management process. An effective e-mail assistant should ideally address all aspects of the process.

Another difficulty in developing a general-purpose e-mail assistant is that users vary widely in the type and quantity of e-mail they receive and their behaviour in managing their mail. Some users receive more than a hundred e-mails per day, others only very few. There have been surprisingly few studies of exactly how users manage e-mail, but it seems most people use folders for filing their e-mails, though some keep all of their mail in the Inbox and rely on efficient search functions to re-locate messages. Of those who use folders, most seem to organize folders based on the content or sender of a message, some organize folders based on the action to be taken in response to the e-mail (reply, delete, etc.), while others again organize folders based on the urgency of the required response. Thus for an e-mail assistant to be widely usable, it must cater for the wide variety of e-mail users and the different strategies they have adopted to cope with managing their e-mail.

In this paper, we present EMMA (*E-Mail Management Assistant*), a personal e-mail assistant that aims to fulfil the requirements of competence and trust, be usable and applicable to a wide variety of users, while also addressing many aspects of the e-mail management process. The heart of the system is a knowledge acquisition technique known as Ripple Down Rules (RDR), which provide both the accuracy of rule-based systems (guaranteeing the correctness and hence trustworthiness of the agent) while simplifying the knowledge acquisition process to such an extent that the user never even has to look at the rule base in order to define or correct existing rules. RDR systems are also *incremental*: the user starts with an empty knowledge base and adds rules while processing examples. Thus in contrast to machine learning algorithms, no training set is required before the system can be used.

In EMMA, individual rules are 'if-then' rules with conditions such as the sender, recipients, subject and keywords/phrases in the content of the message, and with conclusions that may be any combination of a virtual display folder (for sorting messages before the user reads them), a priority (high, normal or low) and an action (read/reply then delete/archive). The user can define templates for automatically replying to messages. In this way, EMMA addresses multiple aspects of the e-mail management process, including sorting, prioritization and automatic reply, using a unified rule-based approach.

RDR systems overcome the major difficulty with rule systems in existing e-mail clients (and other expert systems) – the problem of maintaining consistency in a rule system that potentially contains many interacting rules. However, the onus is on the user to define appropriate rules that classify messages. In this paper, we adopt a machine learning technique (Bayesian classification) to improve the usability of EMMA in three ways: to suggest a folder for filing a given e-mail, to suggest which keywords in a message are useful indicators of a given folder, and to build "interest profiles" that characterize the content of a user's folders.

The organization of this paper is as follows. In Section 2, we give a brief overview of Ripple Down Rules, with classifying e-mail into folders as a motivating example. In Section 3, we describe the EMMA system in more detail, and in Section 4 describe our evaluation of the competence, trustworthiness, and usability of EMMA in a trial of technical users. Section 5 contains a discussion of related work.

## 2. Ripple Down Rules

Ripple Down Rules (RDR), Compton and Jansen [2], are systems of 'if-then' rules organized in a hierarchy of rules and exceptions (exceptions to rules may themselves have exceptions, etc.). An RDR rule base is organized as a tree structure in which each node contains a rule and the children of a node represent exceptions to the rule. RDR systems can be viewed as classification systems when the conclusion of a rule defines a class to be assigned to an example. The use of RDRs supports an incremental approach to knowledge acquisition: starting with an empty rule base, as new examples are encountered, the user adds a rule to correctly classify the example, and when an example is classified incorrectly, the user defines exceptions to the rules that resulted in the misclassification.

There are various types of RDR system. EMMA uses Multiple Classification Ripple Down Rules, Kang *et al.* [4]. With this type of rule base, an example has a classification resulting from multiple conclusions in the rule base. These conclusions are found by following every path in the RDR tree to the most specific node applicable to the example. The classification is then the set of conclusions from all such nodes. For example, consider the RDR tree shown in Figure 1. The root node is a dummy node giving no conclusion. Rules 1 to 5 are the rules in the tree that are not defined as exceptions to other rules, while Rules 6 and 7 are exceptions to Rule 1, etc. Consider an example with features $\{a, b, c, e, f, x\}$ (which might correspond to the sender of a message or keywords in the body of a message). At the top level of the tree, Rules 1, 3 and 5 are applicable to the example. However, Rule 7, which is an exception to Rule 1, also applies to the example. So the most specific rules applying to the example are Rules 7, 3 and 5, giving the classification $\{CRC, Friends\}$. Note how in this example, there are two rules applying to the message that give the conclusion *Friends*, but this makes no difference to the final classification.

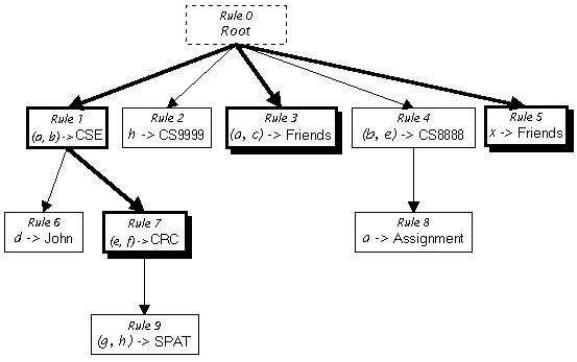


**Figure 1. RDR Rule Base**

An RDR system provides extensive support to the user in defining rules, and this is where RDR systems gain their power compared to traditional rule-based expert systems. The user does not ever need to examine the rule base in order to define new rules: the user only needs to be able to define a new rule that correctly classifies a given example, and the system can determine where those rules should be

placed in the hierarchy. In an RDR system, rules are never modified or removed, only refined or added. Refinement is the creation of an exception rule to correct a misclassification, while addition refers to adding a new rule at the top level of the tree. Rules are always defined in the context of a specific example and in relation to other "relevant" examples, and are not meant to be absolute. To define a new rule "in context," the user has only to identify features of the example that distinguish it from examples that should be classified differently. The RDR system can determine which examples need to be considered by checking the rule base for prior examples that are also covered by the proposed rule. To do this, the system maintains for each rule the example that was used when the user created the rule (called the "cornerstone case"). The cornerstone cases of potentially conflicting rules can be presented to the user as the new rule is being defined, prompting the user to either accept the new conclusion as also applying to the existing cornerstone case, or to identify additional features to further distinguish the current example. The process continues iteratively until the user accepts the new rule and its consequences. This makes the task of defining rules much simpler than if the user is required to define a rule base independently of the context of a specific example (as is required in traditional rule-based expert systems, and in the rule bases of current e-mail filtering systems). In this way, an RDR system can be used by "end users" rather than only by knowledge engineers.

Consider again the example RDR tree shown in Figure 1. A message with features $\{a, c, x\}$ is classified as $\{Friends\}$. Suppose the user wishes to change this classification to $\{ARC\}$, so needs to determine additional features of the example that will lead to the new conclusion. By using the RDR system, the user is certain that the selected features $y$ and $z$ do not result in incorrect conclusions being generated using other rules. For example, as the feature $y$ of the message is selected, the user is shown any cornerstone cases that satisfy the conditions of the new rule, including $y$: note that there may be such cases even though $y$ does not occur in the conditions of any existing rule. The new conclusion $\{ARC\}$ applies to such cases, so the user is prompted to determine whether or not this is the intended behaviour; if it is not, the user must identify additional features (here $z$) that further discriminate the current message from these cornerstone cases. Finally, two instances of the rule 'if $y$ and $z$ then *ARC*' are added as exceptions to Rules 3 and 5 (so that both conclusions are overridden by more specific rules), as shown in Figure 2. The example will now be classified correctly.

In summary, the main advantages of Ripple Down Rules as a knowledge acquisition technique are the high degree of accuracy, the validation of conclusions that the system can provide, and the ease of constructing the rule base. This

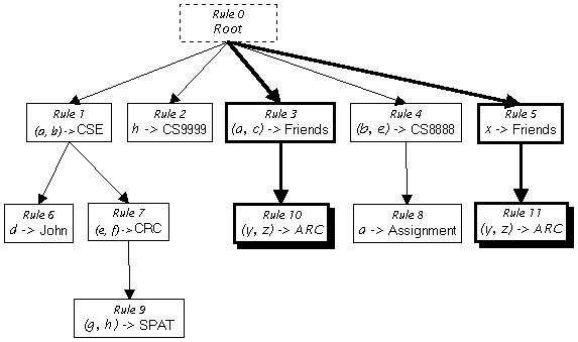


**Figure 2. RDR Rule Base after Refinement**

makes RDR systems suitable in domains where accuracy, explicitness of reasoning, and the justification of conclusions are essential. Typical application domains where RDR systems have been successfully applied are in medical diagnosis, Compton and Jansen [2].

The high degree of accuracy is attained because the user refines the rule base to maintain consistency whenever necessary, so the system always classifies seen examples according to the user's (current) intentions (as the rule base is being constructed, the user can change the classification of previously classified examples by creating exception rules). Total accuracy is not guaranteed, however, because the rules defined by the user are typically over-general, in that rules also apply to unseen examples in perhaps unforeseen ways. However, in practice, since rules apply only in specific contexts, misclassification applies to a relatively small proportion of unseen examples. More usual in practice is that the user's rule base as a whole covers too precisely the set of existing examples, and the RDR system tends to be "conservative" and fails to generalize to unseen examples. Thus in comparison to machine learning systems, RDR systems gain high accuracy (precision) at the expense of coverage (recall) of a set of examples, while typically machine learning algorithms have higher coverage but lower accuracy.

As mentioned above, rules are never deleted from an RDR rule base. To achieve the effect of deleting a rule, a rule may be "stopped," which means that any conclusions that would normally be generated using the rule are ignored, but *not* those obtained from its exceptions (and their exceptions, etc.). One potential disadvantage of this approach is that an RDR rule base can grow to contain a large number of redundant rules, especially in dynamic domains where the user may often want to stop the application of existing rules. However, with the hierarchical structure of an RDR rule base, removing redundant rules is a nontrivial operation, e.g. Wada *et al.* [9], and in practice, has not proven to be necessary.

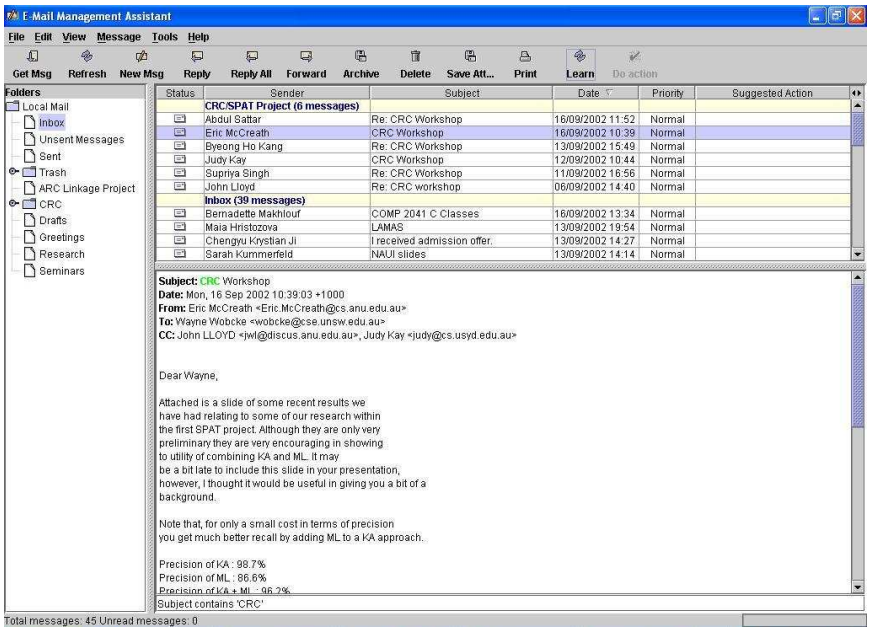


**Figure 3. EMMA Main Window**

## 3. E-Mail Management Assistant

We now describe EMMA (*E-Mail Management Assistant*), whose classification method is based on Ripple Down Rules (RDR). As mentioned in Section 1, the aim of this work is to address as many aspects of e-mail management as possible for as wide a variety of users as possible. The assistant should be competent and trustworthy, and be easy to use. Competence is assured by the use of RDR systems for classification, because of the high degree of accuracy of the constructed rule bases; this leads to a high degree of trust in the system by the users. The EMMA interface is designed to make it easy for the user to view mail, and especially, to define rules.

One reason that Ripple Down Rules are particularly suitable for e-mail classification is the incremental nature of the approach. There is no need, as there is with some machine learning algorithms, for large training sets. The user starts with an empty rule base, and gradually builds up the rule base to classify incoming messages, leading to extensive personalization of the rule base, enabling EMMA to be used by a wide variety of users.

Rules in EMMA (as in all RDR rule bases) are 'if-then' rules applying in specific contexts. The condition part of a rule may be keywords or keyphrases drawn from any of the message headers (*Subject*, *From*, etc.) or from the body of the message (though at present, it is not possible to define conditions based on message attachments). For the rule's conclusion, the user chooses a combination of a virtual display folder (for sorting), a priority (high, normal or low) and an action (read/reply then delete/archive). Using rules with such combined conclusions, rather than having separate rules for the virtual folder, priority and action, leads to fewer rules overall and a more consistent interface for rule building (see Figure 4 below).

The virtual folder is used for sorting and displaying messages to the user, and is based on work in the IEMS system of Crawford *et al.* [3]. The main interface window is shown in Figure 3. Messages in the Inbox are grouped into a number of virtual folders and displayed together in these virtual folder(s). If there is no virtual folder associated with a message, as is the case when no rule applies to the message, the message is left in a virtual folder called 'Inbox'. Virtual folders may correspond to folders for archiving, but this is not essential.[1] Messages may be sorted by date, subject, sender, priority or action, but are still grouped together within virtual folders.

EMMA uses Multiple Classification Ripple Down Rules. Thus more than one rule may apply to a message, and the classification is a set of rule conclusions, i.e. a set of combinations of virtual folder, priority and action. If there is more than one virtual folder for a message, the message headers are displayed under both virtual folders in the Inbox but there is only ever one copy of the message in the Inbox file (so deleting one such message removes both sets of headers from the Inbox display). Within each virtual folder, the priority (and suggested action) of a message is that of the highest priority (high $>$ normal $>$ low) assigned

[1] One of the authors, Wobcke, has a fine-grained set of folders for archiving, but a more coarse-grained set of virtual folders for display.

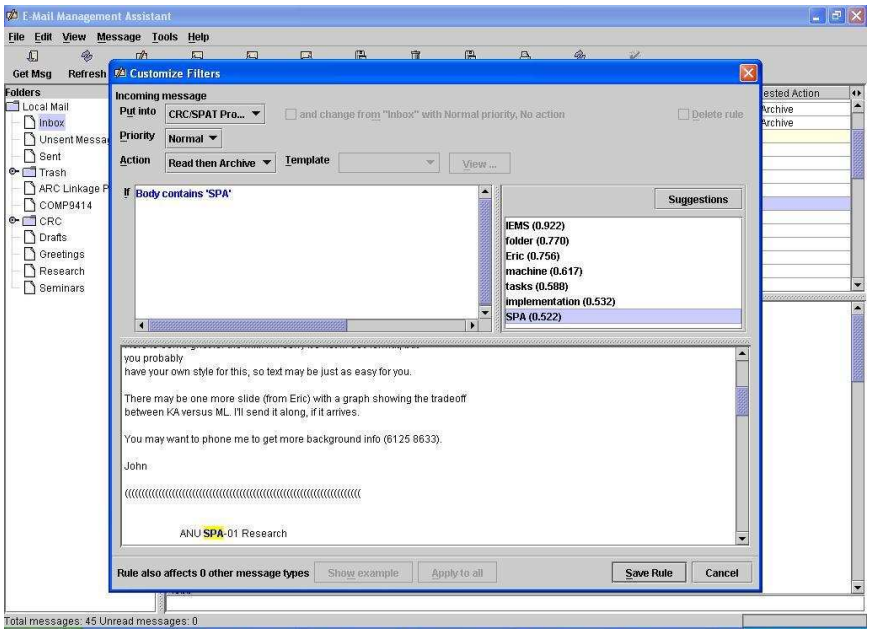


**Figure 4. EMMA Rule Building Interface**

to the message by a rule for that virtual folder; multiple priorities and actions are available to the user by clicking the mouse. The user has the option to display the conditions leading to the given conclusions (keywords and keyphrases are highlighted in the message display), and see the rule(s) that resulted in the current conclusion.

As described above, one main advantage of Ripple Down Rules is the ease with which rules are created. Rules are always defined to classify a given message. When the user wishes to define a rule, the message is displayed in a separate window and the user simply selects various words and phrases in the message headers and body. The rule being constructed is shown in the upper portion of the window, as shown in Figure 4 (in the case of a misclassification, along with any conditions of this rule inherited from its parents in the rule base, as this process refines the rule base by adding an exception rule, as shown in Figure 5).

As the user selects features for the condition part of the rule from amongst the message headers and body, the rule being defined is updated in the upper window, but more importantly, an indication of the number of rules that conflict with the new rule is displayed. Each conflicting rule represents a possibly unintended side effect of the rule, that the present rule will also apply to other previous examples. This indicates that the rule's conditions may be too general and need further specialization. The user can examine typical examples of the conflicting rules (the cornerstone cases, here the messages that were used to create the rules), and either accepts the new classification of the previous cornerstone case, or must further specialize the conditions of the newly constructed rule. The window for viewing cornerstone cases is shown in Figure 5. Using machine learning techniques (see below), EMMA is able to suggest some typical keywords for the intended folder; in Figure 4, the user has selected one such keyword *SPA* that has been incorporated into the condition of the new rule. Any occurrences of the keyword are highlighted in the body of the message.

Finally, the rule will be created and added to the rule base by the system automatically based on the user's chosen options; the user does not have to examine the rule base to determine where the rule(s) should be added. If the new rule is for a message that has been misclassified, the default action is to modify the conclusion, and the system adds exception rules for each rule generating the incorrect conclusion. For a message not classified or where the user wants to define an additional conclusion for a previously classified message, the system adds a new rule at the top level of the rule tree.

### 3.1 Machine Learning in EMMA

One way in which machine learning algorithms can be incorporated into EMMA to improve the coverage of the RDR classification algorithms is by applying one or more machine learning techniques when the rule base fails to classify an example. However, we wanted to evaluate the usefulness of Ripple Down Rules as a method for performing e-mail classification on its own, rather than the usefulness of some hybrid method. Furthermore, machine learning techniques generally rely on extensive training sets

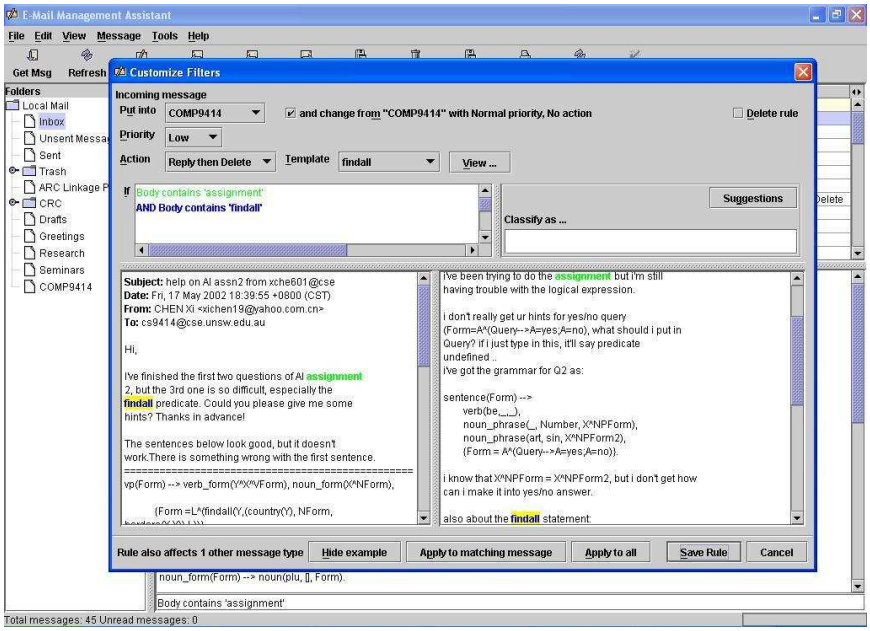


**Figure 5. EMMA Interface for Conflict Resolution**

which were not necessarily available for all our users or whose characteristics would vary widely between users.[2]

Rather than using machine learning algorithms to attempt to improve coverage, we have developed three ways that a machine learning technique (Bayesian classification) can be used to improve EMMA's usability. These are to suggest a folder for filing a given e-mail, to suggest which keywords in a message are useful indicators of a given folder, and to build "interest profiles" that characterize the content of a user's folders. We expected that the most useful of these would be suggesting words from particular messages to be used in rule construction, since rule construction is the hardest part of using the RDR system and this is where Bayesian classification could provide the most help. This is the list of suggestions shown to the user in Figure 4.

The details of the Naive Bayes classification methods are standard, e.g. Mitchell [6]. Data is taken from the user's existing folders. For a given folder $f$, EMMA calculates $P(f|w)$ for all keywords $w$ that occur in any message (in Mitchell's method, all such probabilities are non-zero). The folder $f$ with the highest value for $P(f|W)$, where $W$ is the set of words occurring in a message, is the one suggested to the user for archiving the message; $P(f|W)$ is calculated using Bayes' rule and an independence assumption, so that $P(W|f)$ is the product over the words $w$ in $W$ of $P(w|f)$. Given a particular message which the user has selected to file in folder $f$, the keywords suggested to the user for defining rules are those words $w$ in the message for which $P(f|w) > P(f)$, ranked according to $P(f|w)$. Finally, the "interest profile" for a folder is simply a list of words $w$ occurring in the folder (after removing stop-words) ranked according to $P(f|w)$. Note that the Naive Bayes classification algorithm assumes that the occurrence of each word in a message in a folder is conditionally independent from that of any other word, a simplifying assumption that has been shown to work in practice in many applications. Note also that we have used only single words in the algorithm rather than phrases, for which there is typically insufficient statistical information in a user's mail boxes.

## 4. Evaluation

RDR systems cannot be evaluated in the same way as machine learning algorithms, mainly because of the incremental nature of the approach; there is no clear division between "training set" and "test set." Hence care should be taken in interpreting the results described below, especially taken in comparison to machine learning approaches. In any case, such experiments typically show only general statistics concerning the performance of an algorithm, and do not address the important question of usability.

One major question concerning the usefulness of RDR systems in a given application is the degree to which the user has to continually add rules to cover misclassifications or non-classifications. It is expected that in dynamic domains, users will have to keep adding rules to cover examples not seen previously, and e-mail classification can

[2]The related work of Wada *et al.* [9], which combines inductive learning and Ripple Down Rules, is also directed towards maintaining the consistency of a changing rule base in a dynamic environment.

| User | # Rules | # messages | # classified | # misclassified | % accuracy | % coverage |
|---|---|---|---|---|---|---|
| 1 | 36 | 156 | 134 | 4 | 97.01 | 85.90 |
| 2 | 12 | 259 | 225 | 9 | 96.00 | 86.87 |
| 3 | 8 | 236 | 165 | 7 | 95.76 | 69.92 |
| 4 | 11 | 620 | 460 | 3 | 99.35 | 74.19 |
| 5 | 24 | 367 | 312 | 15 | 95.19 | 85.01 |
| 6 | 131 | 1766 | 1696 | 82 | 95.17 | 96.04 |
| 7 | 6 | 25 | 22 | 1 | 95.45 | 88.00 |
| 8 | 55 | 276 | 258 | 7 | 97.29 | 93.48 |

**Table 1. EMMA User Trial Results**

to a certain extent be considered dynamic in that the type of messages a user receives changes with time. The main question is whether the burden of creating new rules outweighs the usefulness of the classification system. This is a question that can only be answered by asking the users.

We conducted a two week trial of EMMA with 8 technical users in the School of Computer Science and Engineering at the University of New South Wales (not counting the authors, who had been using EMMA for a period of two months prior to the trial). The aim of the trial was to evaluate the competence, trustworthiness and usability of EMMA. More particularly, we were interested in the following aspects of the system.

- The usefulness of virtual folders for displaying messages in the Inbox.
- The accuracy of the classification.
- The ease of the rule building process.
- The degree of use made of different types of conclusion (virtual folders, priorities and actions).
- The usefulness and accuracy of the suggestions provided by the Bayesian classification algorithm.

It was expected that much of the users' evaluations would be coloured by their previous experience with e-mail clients: the two clients most commonly used by the evaluators were Netscape Messenger and Unix mail (pine, elm, mutt). Since some people involved in the evaluation had also been using the filtering systems of other e-mail clients, we also wanted to compare EMMA with their existing e-mail system.

Table 1 shows the characteristics of the rule bases and mail processed by the 8 users in a two week trial (the users are listed in no particular order). Column 2 is the number of rules in the user's rule base – note that because EMMA automatically determines where to add exception rules, the users may not have had to define that many rules (most users had little idea how many rules they had defined). Column 3 shows how many messages were received in the two week period: amazingly, user 6 received 1766 messages in the period, which were mostly junk mail. Column 4 shows how many messages were classified according to the rule base as it was constructed during the period and Column 5 shows how many messages were misclassified (see below for the definition of misclassification). Column 6 shows the proportion of messages classified that were correctly classified (accuracy), and Column 7 shows the proportion of all messages received that were classified either correctly or incorrectly (coverage).

Due to the use of the RDR rule base, EMMA achieves very high levels of accuracy in classifying messages, between 95% and 99% over the trial period (note that during the period users are continually defining rules, affecting both accuracy and coverage). There is no direct feedback from the user concerning misclassifications; EMMA's accuracy is determined as follows. If the message is handled by the user according to the classification or if the user does not handle the message in a way conflicting with the classification, the classification is counted as correct. Only if the message is handled differently from the classification, e.g. a message from one virtual folder is archived into a different folder or the user refines a rule applying to the message that changes its classification, is the classification counted as incorrect. Thus according to this measure, users were able to define rules resulting in a high degree of accuracy.

The coverage attained by EMMA during the trial period is also high, ranging from 70% to 96%. This result is surprising (at least to one of the authors) because of the wide variety of e-mail received, making it difficult to define rules to cover this variety, and because much e-mail is simply read once and deleted (and this author does not feel the need to define rules for this type of message). It is possible that because the trial was over a short period, the mail received by users during that period did not vary much; it would have been desirable to measure coverage over a longer period to see if these high coverage rates were maintained.

Not shown in the table is user feedback concerning the usability of EMMA. All users agreed the rule building interface was easy to use, and that virtual folders provided a useful interface (though some commented that scrolling through a large Inbox was more difficult with this interface, especially when the header of a message appeared more

than once). Few users commented on features such as the creation of semi-automatic replies or prioritization, probably because there was little need for these features during the short trial period.

One aspect of RDR systems that arose relates to the fact that rules can only be refined or added. Users wanted to delete conditions from a rule so as to generalize its application: this is normally done by defining a new rule. The issue is that the old rule persists in the rule base. While this results in no incorrect classifications, unless the rule is stopped, when other rules are refined, the cornerstone case for the old rule may be presented to the user for consideration along with that for the new rule; the user thus sees cornerstone cases that do not need to be examined. Another aspect relating to the rule building language was that some users wanted to define rules with "negative" conditions (e.g. the message does not contain *CRC*), especially in exception rules. There is no reason such rules cannot be defined, but eliciting these conditions would require more complex user input, and for this reason, is not included in EMMA.

## 5. Related Work

There are numerous research prototypes that attempt to address aspects of e-mail management. Only one such prototype, to our knowledge, has been incorporated into a commercial product: SwiftFile in an extension of Lotus Notes. SwiftFile, Segal and Kephart [7], provides the user with three extra buttons for suggesting the folder(s) where a message should be archived: the suggested folders are calculated using the *tf-idf* method. There are three buttons because this provided an acceptable degree of accuracy (73% to 90% in trials). From our perspective, however, the use of three buttons means that SwiftFile can only be used relatively "late" in the e-mail management process, certainly after the user has opened the message. But on the other hand, sorting incoming messages using the first recommendation of SwiftFile does not give an acceptable degree of accuracy (ranging from 52% to 76% in trials).

Cohen [1] compares two techniques for sorting e-mail into folders: the *tf-idf* approach and RIPPER, a rule induction algorithm. Given a training set of messages classified into folders, RIPPER uses keyword spotting to repeatedly add rules into a rule set for each folder until all positive examples are covered while all negative examples are excluded. Initial results seem impressive (the accuracy of RIPPER is between 87% and 94% on varying sized data sets when 80% of the data is used for training and 20% for testing), but it appears this assumes all messages are classified in exactly one of the given folders. Thus these statistics address accuracy, but not coverage of the data, and the accuracy is attained using a "batch" method on a sufficiently large training set.

## 6. Conclusion

EMMA is an e-mail management assistant based on Ripple Down Rules. addressing multiple aspects of the e-mail management process, from sorting messages into virtual folders to archiving messages and creating semi-automatic replies. The RDR approach provides a high degree of classification accuracy, while simplifying the task of maintaining the consistency of the rule base. A Bayesian classification algorithm is used to improve the usability of the system by suggesting keywords for filing messages into a given folder, suggesting a folder for filing a given e-mail, and building "interest profiles" characterizing a user's folders.

## Acknowledgements

This work was funded by the CRC for Smart Internet Technology. We would like to thank Pacific Knowledge Systems Pty. Ltd. for the use of their RippleDown™ software for implementing the RDR rule bases, Eric McCreath for supplying a version of the IEMS software that included an interface for displaying sorted messages, and finally the trial participants for their time and helpful feedback.

## References

[1] W. W. Cohen. Learning Rules That Classify E-Mail. In *Machine Learning in Information Access: Papers from the 1996 Spring Symposium*, pages 18–25, 1996.

[2] P. Compton and R. Jansen. A Philosophical Basis for Knowledge Acquisition. *Knowledge Acquisition*, 2(3): 241–257, 1990.

[3] E. Crawford, J. Kay and E. McCreath. IEMS – The Intelligent Email Sorter. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 83–90, 2002.

[4] B. Kang, P. Compton and P. Preston. Multiple Classification Ripple Down Rules: Evaluation and Possibilities. In *Proceedings of the $9^{th}$ AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 17.1–17.20, 1995.

[5] P. Maes. Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7): 31–40, 1994.

[6] T. Mitchell. *Machine Learning*. McGraw Hill, New York, NY, 1997.

[7] R. B. Segal and J. O. Kephart. Incremental Learning in SwiftFile. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 863–870, 2000.

[8] S. Singh and A. Ryan. A Design for the Effective Use of Corporate E-mail. Research Report 27, Centre for International Research on Communication and Information Technologies, RMIT University, 2000.

[9] T. Wada, T. Yoshida, H. Motoda and T. Washio. Extension of the RDR Method That Can Adapt to Environmental Changes and Acquire Knowledge from Both Experts and Data. In *PRICAI 2002: Trends in Artificial Intelligence*, pages 218–227. Springer-Verlag, Berlin, 2002.