Proceedings of the 5th WSEAS Int. Conf. on MULTIMEDIA, INTERNET AND VIDEO TECHNOLOGIES, Corfu, Greece, August 17-19, 2005 (pp11-16)

Maintaining Rule Friendly for Email Management

Alex C. P. Lai
Department of Computer Science
University of Sydney
Australia

Abstract: Every regular email user is well aware of the problems for managing large numbers of messages in his/her mailbox, because of the large amount of email that people receive every day. As we knwo the most users feel difficult to maintain filter rule for email classification problem, even expert users still have to put some effort on the task. It will be valuable if users can have a smart system to help them maintaining email filter rule. We thus integrated: machine learning for defining the rules, scrutable model, and programming by demonstration, called the Scrutable Rule Interface, and added them into a part of IEMS, to show how they can encourage and assist users to manage filter rule for email management. We describe a practical assistance interface and present empirical results that found general and expert users feel more confidant to infer thier own rules or apply other mechanizms used to infer the filter rules.This improved performance to acceptable levels.

Key-Words: Scrutable Personalization, Email Management, User Model, Machine learning

1 Introduction

This paper describes the Scrutable Rule Interface which has the broad goal of improving our understanding of how to build systems which can assist users in managing filter rule for automatic email classification. 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 supervisor may be filed into your "supervisor" 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 Messenger and Microsoft Outlook Express. The rule is combined with a list of keywords or Strings appearing in the natural email messages. To file email messages into right folder, the rule 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 challenge part of rules is that the process of a rule is cognitively demanding and there is a real, potentially unacceptable risk of misfiling mail. Previous works, shown users seem to avoid customizing software. The most recent work, study of user's management of email, the authors observe "Most of ours 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" [6] [11].

The motivation for maintaining rules is based on first, a belief these rules will relatively easy for end users to understand, review, accept and modify [], second, a suspicion that machine learning methods alone are not an adequate solution for categorization problems of this type. It seems likely that instead some mix of automatically and manually constructed classifiers will be necessary to account for the fact that both the user's interests and the distribution of message change (some times quite rapidly) over time. In many cases, the users themselves might have a better idea of the rules that would be appropriate for the applications they would like to build, so it would useful for them to be able to specify new rules. For instance, at the time of this writing, the rules above may be accurate for messages I have received over the last few months; however at some point it will certainly become appropriate to modify it by replacing "Assignment No: 05" with "Assignment No: 06" and "Assignment No: 5" with "Assignment No: 6".

Our solution is to build a Scrutable Rule Interface added them into a part of IEMS [4], base on the technology of machine learning methods for defing the rule, Scrutable Model [4], Programming by Demonstration [8] that reduces the cognitive burden and the time required for easily understanding and customizing rules, to solve email classification into folder automatically in sort time. As we know rules make decisions based on a small number of keywords. Rules do not base classification decisions on word frequency, only on the presence or absence of a word. A problem with the Scrutable Interface has always been how to represent the rules to users, and how the users can come to feel in control the whole processes is becoming our major problem.

This paper describes the approach to solve the above problem. In section 2, we describe related work and then, in Section 3 we introduce our system approach to maintaining learning rules and Section 4 reports evaluation of that approach. Section 5 concludes with a discussion of the results.

2 Previous Work

Most of the current systems have used simple rule based differencing for their generalizations. For example, the early Peridot system, developed in 1987 to create widgets by example uses about 50 hand-coded rules to infer the graphical layout of the objects from the example [9]. Each rule has three parts, one for testing, one for feedback, and one for the action. The test part checks the graphical objects to determine whether they match the rule. For example, the test part of a rule that aligns the centers of two rectangles checks whether the centers of the example rectangles are approximately centered. Because these rules allow some sloppiness in the drawing, and because multiple rules might apply, the feedback part of the rule asks the user whether the rule should be applied. For example, Peridot might ask something like "Do you want the selected rectangle?" If the user answers yes, then the action part of the rule generates the code to maintain the constraint. The subsequent systems have used similar mechanisms, though often without the explicit list of rules we used in Peridot. For example, Tourmaline, which formats documents from example, contains rules that try to determine the role of different parts of a header in a text document, such as section number, title, author, and affiliation, as well as the formatting associated with each part. The results are displayed in a dialogue box for the user to inspect and correct.

Results have shown PBD has very high performance to solve classification problems based on rule-based inferencing, from different users (especially novices) by a series of very simple actions. As we know the purpose of PBD's characteristic is that it is easily understood. This characteristic led us alternative approach to encourage the users to solve email classification problem.

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 categorization. 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 any instance, it is desirable that any learning algorithm should produce useful results quite quickly, with small amounts of data: in our case, the learning is for a single user's filtering preferences and it is desirable that rules for automating this should be learnt from small numbers of example.

We know the email management task is very sensitive, and not allow any errors occur. We believed that users must spend some time on observation and test of the system, to see the result of how well the system are and what the system can do. The classification processes should be scrutable: the user should be able to scrutinize the whole process so that thy feel in control of the filtering rules or other mechanisms used to manage the email account.

Another important aspect of the control panel should be constructed as easy to use and control interface, adopted by all users. There are only one technology can reach that goal, called Programming by Demonstration technology. The user only required some experience of drag and drop without learning any computing skills.

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 recognize 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 categorization for different mail classes. We know that different people use quite different mail management strategies, as noted, for example in [6]. 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 [12], using a Naïve Bayes approach achieved accuracy of 94%. Sahami *[16]* 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 *[1]* compared Naïve Bayes with a keyword approach. The keyword approach uses the keyword patterns in the anti-spam filter of Microsoft Outlook 2000 (which they believe to be hand constructed). They reported the keyword approach achieved precision of 95% and the Naïve Bayes approach 98%. On recall the corresponding performance was 53% and 78%. Androutsopoulos *[1]* also explored a memory based learning approach IiMBL (a simple variation of the K-Nearest Neighbour) showing similar performance to a Naïve Bayes classifier. Provost *[14]* also evaluated Naïve Bayes, comparing it with the RIPPER algorithm, showing 95% accuracy after learning on just 50 emails while RIPPER reached 90% accuracy only after learning on 400 emails.

One fairly strong result is described by Cohen *[3]*. He used the RIPPER learning algorithm to induce rules and reported 87%-94% accuracy. He also explored TF-IDF, and achieved 85%-94%. He observed that the rule based approach provided a more understandable description of the email filter. The iFile naïve Bayes classifier [15] was made available to several users to test on their own mail and this gave 89% accuracy. Grutlag (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.

The more general categorization task achieves weaker results than the levels achieved for two-category spam filtering employed by various researchers. Agent [2] 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 datase where the standard IR approach had 91% accuracy.

Golbeck and Hendler (2004) have addressed out previous researcher how to fight with Spam, including whitelist and social network based filters – are being investigated to further improve on mail sorting and classification. They present an email scoring mechanism based on a social network augmented with reputation ratings, as well as an algorithm for inferring reputation ratings between individuals and demonstrate through experiments that it is accurate based on current data. In addition, they provide a mail application, called TrustMail, to show how they may be used in combination with other techniques for sorting and filtering mail [7].

Tretyakov (2004) has presented some of most popular machine methods (Bayesian classification, k-NN, ANNs, SVMs) and of their applicability to the problem of spam-filtering. The effort could help people to understand the learning algorithm, even a reader not familiar with them before. Later, the author makes some trivial sample, take them to compare with standard spam corpus. Finally, some ideas come out to construct a practically useful spam filtering using the discussed techniques, and suggested the only reliable way of filtering spam is by creating a set of rules by hand strongly [20].

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 summarized above seem very high for any realistic scenario where the user might have modest numbers of mail items in many of their mail folders.

The work described above does suggests that automated classification should be able to operate usefully in helping users create classification mechanisms for their email. 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.

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 [17], 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.

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 *[13]* who reported a study where users were asked to assess their preferences for

different approaches for representing email filtering rules.

Our work is similar to previous work as PBD approach, and machine learning for defining the rules. The significant part of PBD is allowed user to explore the whole process and adjust the rules defined by the machine learning such as TFIDF, Sender, Keywords, DTree, and Naïve Bayes. After processed, it performs prediction as the presence or absence of a word.

3 System approach to maintaining learning rules.

Previous research work, indicated that various learning algorithms does email classification, results show that is not well in general classification, much better in Spam filtering. Significantly Cohen has concluded the TFIDF is much readable when machine does learning rule, and the performance is much better than other machine learning algorithm for automatic email classification, compared with other machine learning algorithm.

We have combined the Scrutable Model, Machine learning base, and Programming by Demonstration. First, Scrutable Model is a kind of intuitive interface, which communicate with machine learning base (keyword, sender, TFIDF, Dtree) then bring rich information to user such what system can do or how well system does. It will encourage user to understand and adjust filter rule at right time. The main goal of the PBD technology is to construct an easy to use interface, only required some experience of drag and drop without learning any computing skills.

Figure 2 shows an example of a Scrutable Rule Interface is a part of IEMS [4]. The very top button enables users to solve some FAQs. The first textfield indicates the folder name, which has been predicted by the system, such as the ‘hardware’ folder. The second textfield indicates the destination folder such as ‘hardware’ folder, which has been done by users. In addition, the first textarea shown the actual mail message particular with some highlighting keywords/phrases as green color, according to a list of keywords/phases showing from the second textarea at very bottom. This screen allows users to add/delete keywords/phases by simple click on add/delete button. This effort improved performance to recognizable levels.

Once a user has read a message, there are two possible courses of action courses of action. If they are happy with the classification, they can simply click on Archive button. This is at the top left of the screen. This moves the message into that folder. In the case of the current message shown in Figure 1, the archive button would move it to the ‘atheism’ folder. At the same time, User might wish to see the reason of why a message is classified into the ‘atheism’ folder. The system allows user simply click on the

Fig. 1 the IEMS

Fig. 2 the Scrutable Rule Interface

‘atheism’ folder, and select on the message, which might want to see. (see Figure 2) It will explain the rule particularly in highlighting some keywords.

The other possible case is that the user is not happy with the classification. In that case, the user simply selects the MoveTo button followed by the name of the folder in the left panel (see Figure 1). At the same time, a scrutable interface prompts out, allows user to scrutnize the rule particularly in highlighting some keywords, and waiting user to adjust rules if they feel confident (see Figure 2).

This interface should help users to maintain the filter rules, as well as encourage users to get involved with email classification tasks. If the system makes the correct classification, the user simply accepts it with a single click. If the system is wrong, the user

does the sort of classification task (particular in adjusting rules) they would have had to do anyway. This should significantly reduce the cost of creating a filter rule while improving the accuracy for email classification.

4 Empirical Results

Below we describe the experiments that have been done so far. First, we compared experience and non-experience users to see whether they could undertake the same task (creating a rule) or not. Second, we want to see whether reconstructing the rules was a difficult task or not should disaster occur. Details as below:

4.1 Experiment 1

We conducted this experiment in 20 minutes, setting up 10 computers with fully installed iems system, at a University computer laboratory. We arranged two groups called Group A, & Group B. Each group had 5 members, all students from university. Each member has to make more than two folders. We asked both groups to create a rule and compared how much time they took to customize a rule by using the same amount of messages (50 messages) from newsgroup.

Group A had some experience in customize rules for email classification and group the emails into folders. Group B has no idea about rules, but they knew how to group the emails into folder. From our observation, we found that most Group B's member only run the system directly without thinking, and they found that the system help them to predict a folder for each email. They could identify error of prediction from some emails, and the system enabled them to see the result of how system has been done so far, especially in highlighting some of most important keywords in green, based on the machine learning result and what the user input manually. Some users tried to add/delete some of keywords and clicked on the confirm button. They tried to test the system in order to see the different result produced.

From the results, we believe that most Group B's members were able to create a rule in an average of 3 minutes. See Figure 4. Group A had good results as well, they could customize a rule on average in under 1 minute. It is very clear that both groups can perform and feel confident in customizing a rule for email classification in very a short time. In addition, we found that Group A's members were able to create more complex rule, not like Group B's members were only created a simple rule by using small number of email messages.

After this test, we gave them a questionnaires. The feedback was very positive. We found the Group A members were very happy with the Scrutable Rule Interface as they felt it was easy to turn the rules, and believe they could customize a new rule in very as short time. Group members enjoyed creating a rule, because they could use their natural ability to identify error prediction easily. Meanwhile, they could change the rules as naturally without any stress.

4.2 Experiment 2

This experiment was conducted in 20 minutes. First we asked 5 users from university lectures and 5 users from university officer to join our experiment. These students had been attacked by computer disaster, such as viruses, hard drive fault and operating system failure or other effects. They also had experience using Microsoft Outlook Express filters and know how to use rules to file email into folder automatically. We asked them to use our new iems system instead of their current system to maintain filter rule, while were created more than two folders.

After 20 minutes, we found very positive answers. They found iems system very powerful in creating rules in a very short time. They didn't feel that the system required much effort to use and they believed that if a disaster should happen again, they would not worry about recreating the rules. One thing they worried about was how to back up the email files and it would seem that this is becoming another important task.

5 Conclusion

The Scrutable Rule Interface approach is an easy-to-use personal assistant that helps users create a rule in a way more natural to them to solve their email classification problems. Scrutable Rule Interface approach makes very few demands on users if apply to another email client; they have approach makes very few demands on users; they have nothing extra to learn when creating the rules and the only thing required is their natural ability. Users can identify the wrong prediction easily and move to the mail to the correct folder. Furthermore, it pops up with an interface to provide details (especially highlighting some keywords) and to ask them to modify it if necessary. In the future, if the system detects a similar email coming in, it will predict a folder for this email automatically. Users are only required to do a final confirmation such as clicking on Achieve button. This improved performance to acceptable levels.

The experiment results are very positive, as previous discussed. Experience and Non-Experience users can do the same task after 3 minutes. Another experiment results, experience users are worry about computer disaster, such as viruses, hard drive fault and operating system failure or other effects. After 20 minutes, they believe this is the right technology to assist them in recovering rules in a very short time. We found the more general categorization are created the performance is not effect. This amounts to a significant qualitative enhancement that is likely to encourage users to file their mail using email filter by naturally ability. While Scrutable Rule Interface, has welcome for any other email client to add-on. It is easily be used to organize other types of electronic documents such as disk files, audio, book-marks, recordings, and other text-based documents that are placed into a hierarchy of folders.

4 Acknowledgements

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

References:

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, & C. Spyropoulos, An experimental comparison of naïve Bayesian and Keyword-based anti-spam filtering with personal e-mail messages, *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval in*, 2000.

[2] G. Boone, Concept features in re:agent An intelligent email agent, *Second International Conference on Autonomous Agent*, 1998.

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

[4] E. Crawford, J. Kay, & E. McCreath, Automatic induction of rules for email classification. In *Proceeding of the Sixth Australiansian Document Computing Symposium, coffs Harbour, Australia*, 2001.

[5] A. Cypher, *Watch What I Do: Programming by Demonstration*, MIT Press, Cambridge, Mass, 1993.

[6] N. Ducheneaut, & V. Bellotti, Email as habitat: an exploration of embedded personal information management, *University of California, Berkeley*, 1996, pp. (8)18-25.

[7] J. Golbeck, & J. Hendler, Reputation Network Analysis for Email Filtering, *University of Maryland, College Park. MINDSWAP. 8400 Baltimore Ave. College Park MD 20742*, 2004.

[8] T. Lau, & D. S. Weld, Programming by Demonstration: An inductive learning Formulation, *Intelligent User Interface*, 1999, pp. 145-152.

[9] B. Myers, Creating user interfaces using programming by example, visual programming, and constraints, *ACM Transact*, 1990, pp. 143-177.

[10] K. Mock, An Experimental Framework for Email Categorization and Management, *Research and Development in Information Retrieval*, 2001, pp. 392-393.

[11] W. Mackay, Triggers and barriers to customizing software, *CHI'91 Conference on Human Factors in Computing System, New in Computing System, New Orleans, Louisiana*, 1991, pp. 153-160.

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

[13] M. Pazzani, Representation of electronic mail filtering profiles: A user study, *Proc. ACM Conf. Intelligent User Interface. ACM Press*, 2000.

[14] J. Provost, Naïve-bayes vs. rule-learning in classification of email, 1999.

[15] J. Rennie, Ifile: An application of machine learning to e-mail filtering, *KDD-2000 Text Mining Workshop, Boston*, 2000.

[16] M. Sahami, s. Dumais, D. Heckerman, & E. Horvitz, A Bayesian approach to filtering junk email, *AAAI-98 workshop on learning for Text Categorization in*, 1998.

[17] R. Segal, & M. Kephart, MailCat: An Intelligent assistant for organizing e-mail, *Proceedings of the Third International Conference on Autonomous Agent, Seattle, WA*, 1999, pp. 276-282.

[18] D. Smith, A. Cypher, & L. Tesler, Novice programming comes of age, *Communication of the ACM*, 2000, pp. 43(3):75-81.

[19] T. Scheffer, Email Answering Assistance by Semi-Supervised Text Classification, *In Intelligent Data Analysis*, 2004.

[20] K. Tretyakov, Machine Learning Techniques in Spam Filtering, *Data Mining Problem-oriented Seminar, MTAT.03.177*, 2004, pp. 60-79.