

Almost junk: classifying public announcements for user communities

Florian Verhein, Judy Kay, and Irena Koprinska
Smart Internet Technologies Research Group,
School of Information Technologies,
The University of Sydney,
NSW 2006 Australia.
Email : {fverhein,judy,irena}@it.usyd.edu.au

Eric McCreath
CRCSIT
Department of Computer Science,
The Australian National University,
ACT 0200 Australia
Email: ericm@cs.anu.edu.au

Abstract—This paper describes our work towards building a smart personal assistant that helps users deal with the ever increasing volume of email. To this end, we filter incoming messages for the user, organising messages in ways that enable the user to deal with the information effectively. We describe an initial corpus of public email built for experiments on learning to classify email. We also report initial results of experiments based on that corpus. These explore the question of how well a classifier can perform the task of classifying a user’s email into categories that reflect the users’ thinking about that email. We also report initial experiments aimed at answering a second question: if a learner has been trained by one user, how effective is it for another user? That is, can we share information about classifications of messages improving collective performance?

I. INTRODUCTION

Information overload poses critical challenges, as increasing amounts of information become available through the WWW and as large amounts of it are delivered to users. At present, one of the important ways that information is pushed to people is via email. With the growth of novel interfaces and their ubiquity, we can expect to see many more forms of information push, such as SMS, mobile phone messages. This makes it critical to improve our understanding of how to help people manage email effectively: this is a testbed for a future where many forms of information are pushed to people through the web and networks.

We have been exploring the combination of machine learning and other classification approaches in conjunction with a novel email interface [11], [9]. The novelty in the interface is that it attempts to presort the user’s inbox into the categories that the user normally sorts mail into. If this is successful, it means that the user can then deal with related sets of email together. This offers a substantial benefit in reducing context shifts and in enabling the user to deal with more important tasks more effectively. It also has the potential to speed up the process of moving the mail from the inbox to the appropriate folder.

One of the central questions in this work is just how well a classifier can perform on this task of classifying a user’s email into the categories that they choose to structure their thinking about email. A second question is: can we improve

the learner’s performance by exploiting people’s community membership. It is these questions that we have been exploring.

The answers are likely to be different for different people and for different aspects of email classification, since there is considerable evidence of quite different email management strategies for different users [7], [9], [8]. As a first step, we needed to define a suitable collection of classified email that could be used in a series of experiments. Ideally, we would have been able to draw upon an existing corpus but none was available. Therefore we have been building a corpus, as we describe in Section 2. Section 3 describes our experiments to explore how well standard machine learning tools can perform on the task of predicting user’s classifications and which machine learning approaches appear to perform well on this task. In Section 4, we consider prediction of classifications based upon training the learner on one user’s data and assessing how this might perform in predicting other user’s classifications.

II. THE E-MAIL CORPUS

There has been considerable work on machine learning for email classification. A series of particularly successful experiments *SpamCop* [13], Sahami et al. [16], Katirai [10], Provost [14] has given excellent results in distinguishing junk mail, with a range of different classification techniques. Androutsopoulos et al. [2] built and made publicly available two large corpora for junk email filtering (PU1 and LingSpam) that have been widely used as a benchmark. It was found that Naive Bayes (NB) outperforms both RIPPER and a keyword-based filter similar to those used in Microsoft Outlook [1] Sakkis et al. [17] combined a Naive Bayes and k nearest neighbor by stacking and found that the ensemble achieved better performance. Carreras et al. [4] showed that boosted trees outperformed decision trees, Naive Bayes and k-nearest neighbor. A neural network-based approach [5] has been shown to outperform the above mentioned classifiers. All these papers report extremely high accuracy, precision and recall on the junk mail classification, typically better than 95% on all of these measures. Since the junk mail classification task is relatively easily appreciated (though not well or stably defined), we might consider these results to be comparable. In that case, they indicate that a range of different machine learning

techniques achieve similar high performance for filtering junk mail.

As one would expect, the more general classification of email has had poorer performance. If one explores the considerable amount of work on this task, the collection of results are hard to reconcile and compare. For example, Cohen [6], reports 87%- 94% accuracy in one experiment, 85%-94% in another whereas Rennie [15] reports 89% accuracy, Brutlag and Meek [3] 70% to 90% with one learner, 65% to 90% with another and 67% to 95% with yet another. Notably the range of results reported depends upon the particular mail corpus used. In our own work [7], [9], we compared the effectiveness of different learners for different users, each classifying their own email. However, these results do not give a good indication of what we might reasonably expect of a classifier for this task as it is almost impossible to compare results derived from different data sets and using different experimental and evaluation setup. It is timely to build a corpus of classified email that can be used in experiments which evaluate different machine learning approaches. It is important that we move beyond just the well-defined area of junk mail to a task that is more typical of that we might like to delegate to a smart personal assistant.

A. Collection of the email

Our first task was to collect suitable email messages. We asked several users to collect mail that was essentially a public announcement in some form. Participants were asked to save and make available to us *public* email about events. We asked for mail such as the following: calls for papers, notices of seminars, other departmental events. These constitute a collection of mail that is quite distinct from junk mail in that at least some of it is of considerable importance to the recipient. At the same time, it is different from purely personal email and it constitutes an interesting class of mail: users receive considerable amounts of this mail and some of it is of varying degrees of usefulness while the bulk is not of interest. This paper reports our experiments on one of these collections: it was by a single over 7 months in 2002.

We processed each collection, deleting all headers except From, Date, Subject and our own special identifier header. We also removed all binary types of content as well as extracting actual text from html documents. Effectively we strip the messages back to this main headers and the text of the body. Finally, we removed any emails that we thought were 'not public enough' manually. We purposely left all duplicates and near duplicates in the set since they are an part of the authentic data set.

B. Collection of classifications

It is important that our email corpus should enable us to evaluate the effectiveness of classifiers for different people. Accordingly, we asked several users to classify the email into five categories. These are shown in Table I. The reason for the two classes of duplicates was to ensure that the classification of Important and Interesting were unambiguous: if a person

Category	Guideline
Important:	I definitely wanted to see this.
Duplicate Important:	As above but I have already seen it.
Interesting:	I want to see but not as important as the above.
Duplicate Interesting:	As above but I have already seen it.
Uninteresting:	prefer not to see.

TABLE I
THE FIVE CATEGORIES USERS CLASSIFY THE EMAIL INTO.

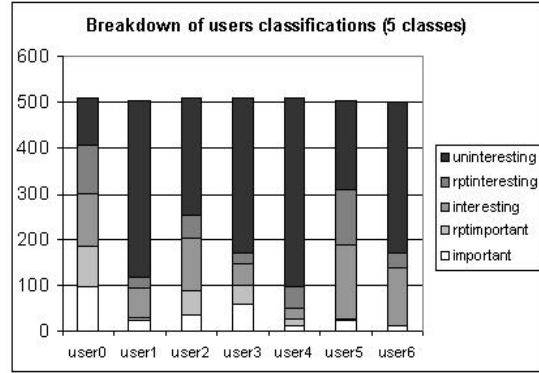


Fig. 1. Breakdown of users classifications into the 5 original classes. User0's classifications are distributed randomly and uniformly across the 5 classes.

is classifying an email that they have previously classified as Important, they may decide to class it as Uninteresting because they had already seen it. We did not want this to happen. Users were free to do this classification task in any way they wished. Six users have classified the first email set. Their classifications were used in the experiments that we now report. Figure 1 shows the distribution of mail items in each classification across these five categories. Figure 4 shows the corresponding distribution where these are folded into three classes: important (including duplicates), interesting (including duplicates) and uninteresting.

C. Feature Selection and Dimensionality Reduction

We explored two different sets of features: the sender and the union of the email body and subject.

Sender. : The size of the feature vector was 159 and the individual features corresponded to the email addresses of all the different people who sent mail. A binary representation was used.

Body and subject. : A bag of words representation with tf-idf weighting was applied. First, all unique words in the body and subject of the email corpus were identified (approximately 13 0000). A feature selection mechanism was applied to reduce dimensionality: words that appear in fewer than X documents were discarded. The value of X was empirically set to 60 and this reduced the number of unique words to 320. Each email was then represented by a vector that contains the tf-idf weighting of the selected features.

Classifier	Description
IBK	K nearest neighbor. We varied K over the range 1 to 25: 1,5,10,15,20,25.
ZeroR	Zero Rule. Predicts the majority class. Used as baseline.
OneR	One Rule. One level decision tree expressed as a set of rules testing the value of one attribute.
NaiveBayes	Statistical method. Assumes independence of Attributes. Uses conditional probability and Bayes rule.
J48.j48	C4.5 algorithm. A decision tree learner with pruning.
Decision Table	A simple decision table majority classifier.
Id3	Id3 decision tree induction algorithm.
Bagging	We used 5 bags with either 66% or 100% resampling on various classifiers.
Boosting	We used the AdaBoostM1 algorithm on various classifiers.

TABLE II
THE ALGORITHMS USED FROM THE WEKA REPOSITORY.

III. CLASSIFIERS USED

Table II summarizes algorithms we employed. We used their WEKA [18] implementation.

IV. EXPERIMENTS

A. Learning to model user interest

The goal of the experiments was twofold: to gain a sense how the machine learning algorithms perform overall using the two set of features, and to compare performance across users.

We used 10-fold cross validation and calculated accuracy, precision, recall and an average F measure which weighted recall by a factor of 2 and precision with a weight of 1. This is in contrast to the usual F1 measure which weights them equally. We consider that in this domain recall is more important than precision and so chose a measure to reflect this. This is particularly true in our email interface because the goal is to provide the user with related items. This makes recall more important in any trade-off between recall and precision, since it is preferable to have some extraneous items than to miss ones which belong in the category.

To provide a benchmark for the other classifiers, we included ZeroR. It is a simple classifier that simply predicts the majority class (that was always class uninteresting). If a classifier performs worse than ZeroR this indicates substantial overtraining. We also included a random user, User00, which classified all messages into important and repeat, interesting and repeat, or unimportant with equal probability. This was another benchmark that helped us to distinguish spurious results. In particular, it gives an indication of how much care people took in classifying the email, and secondly it gives an idea of the level of performance required in order to deduce that there is a pattern present.

B. Overall performance

Figure 2 and 3 show the classification accuracy when sender and body&subject were used, respectively. Decision trees and 1-nearest neighbor were found to be the best classifiers on both features. Based on Sender the best accuracy ranged from

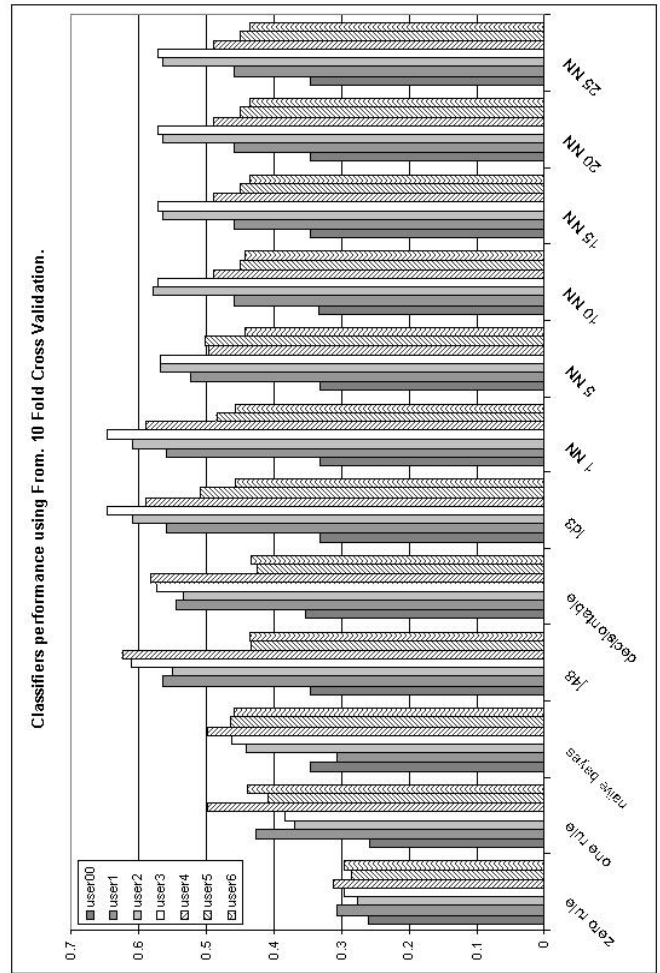


Fig. 2. Accuracy on different users using Sender.

65% on user 3 to 45% on user 6. The use of the bag-of words representation of the e-mail's body and subject increased the accuracy with almost 10% (ranging from 75% on user 3 to 52% on user 6). Thus, the content of the email is more important than the sender when classifying email.

C. Comparison across classifiers

All algorithms performed better than ZeroR and their accuracy on users 1-6 was always considerably higher than on user 00. The one-level decision tree 1Rule performed relatively well compared to more complicated classifiers such a Naive Bayes. This confirms that it is worth trying simple classifiers first as they sometimes work surprisingly well.

The good performance of nearest neighbor and the unsatisfactory performance of Naive Bayes are consistent with the results of Yang and Liu [19] on Reuters data.

The lower results of Naive Bayes when Sender was used can also be explained by its problems with sparse data. Recall that in this case each attribute is an email address. Thus each email is represented by a binary feature vector with only one attribute having a value of 1. Thus, when the conditional probabilities are calculated (for example the conditional probability that

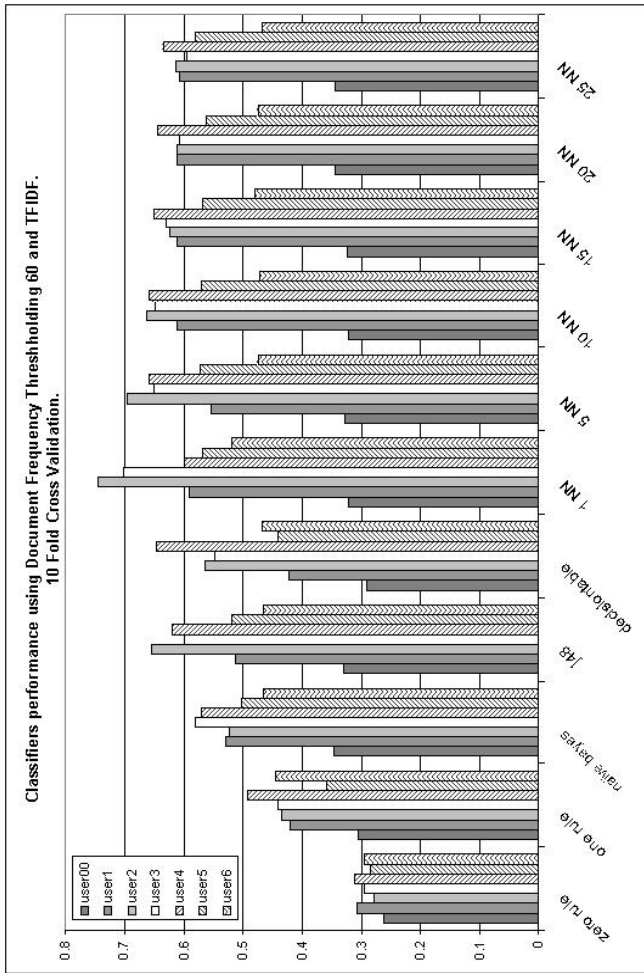


Fig. 3. Accuracy on different users using e-mail body and subject.

address X was the source of the email given that it is in class Y), these are very small. This is especially important as it is very likely that some attributes do not occur in conjunction with every class value. (For example, a user might always find emails interesting or important if they come from X , and never uninteresting). If the Laplace estimator was not used, this would cause problems as it allows zero probabilities to creep in and hold a veto over the other probabilities. We used the Laplace estimator, but we might be overcompensating by giving what would be zero probabilities a value similar to the other probabilities. This might explain why Naive Bayes performs significantly better when tf-idf is used for feature representation, as tf-idf will cause higher conditional probabilities in general, as it is very likely, especially when using frequency thresholding for feature selection, that most of the attribute values will be non zero.

Decision Trees are one of the top performers on both feature sets. In contrast to Naive Bayes, they do not give equal weighting to each attribute but have an in-built mechanism for attribute selection based on the information gain. Duplication of an attribute will make no difference to the learning process

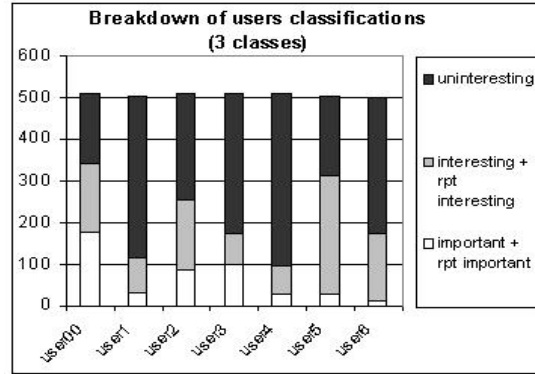


Fig. 4. Breakdown of users classifications into the 3 (amalgated) classes. User00's classifications distributed randomly and uniformly across the 3 classes.

because if one copy is split on, the other copy will have a consistently low information gain at each subsequent (down the tree) split. Both decision trees and decision table do a little better when TFIDF is used.

With K nearest Neighbours, it was found that increasing the value of K does not improve performance. This was surprising as higher K generally work better for noisier data and we had assumed that our data was quite noisy. Furthermore, the values seem to stabilize quite rapidly. See Figures 2 and 3. Overall, this scheme performed very well for all users.

D. Comparisons across users

User2 and user3 consistently had better results throughout most of the classifiers. In fact, the values for both users are very similar for all of the classifiers. This suggests some sort of similarity between these users.

Overall, the distribution of performance scores between users seems to be very similar over most of the classifiers. This seems to suggest that there are no classifiers that are suited to one person only. Classifiers that do well do well for all users in general. This is good news as it should allow generic software to be developed based on one, or at most a few, classifiers.

E. Effect of the class distribution on classifier performance

As Figure 4 shows, the distribution of classes between users varies substantially. It is imperative to explore the effect that this might have on the performance of the classifiers. For example, user1 and 4 have very similar distributions, and most of the classifiers produce similar accuracy results (apart from using Naive Bayes and decision trees (j48) on Sender and also decision table and j48 on the tf-idf features). The effect on the other performance measures, though, seems to be affected more by the class distributions.

We found that recall fell uniformly for all users as K was increased in K nearest neighbors when using TFIDF, see Figure 7. Precision on the other hand showed a significant increase for user1 and user4 while the others increased only slightly or not at all. This can be seen in Figure 6.

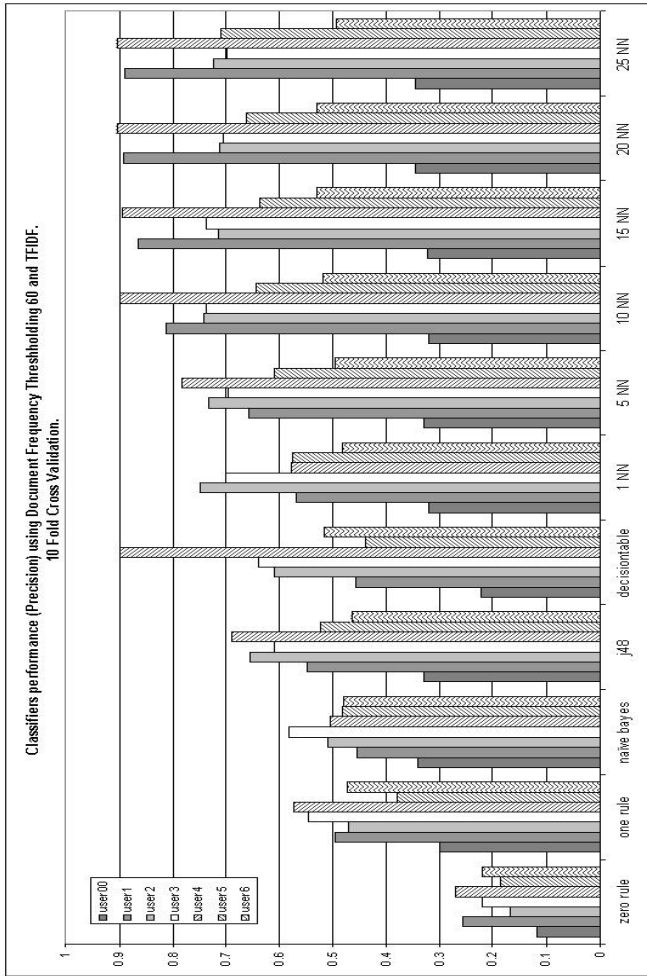


Fig. 5. Performance measured using precision of Classifiers on different users using Document Frequency Thresholding 60 and TFIDF. 10 Fold Cross Validation.

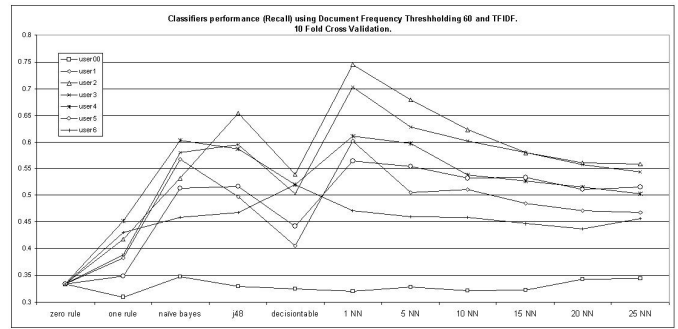


Fig. 7. Performance measured using recall of Classifiers on different users using Document Frequency Thresholding 60 and TFIDF. 10 Fold Cross Validation.

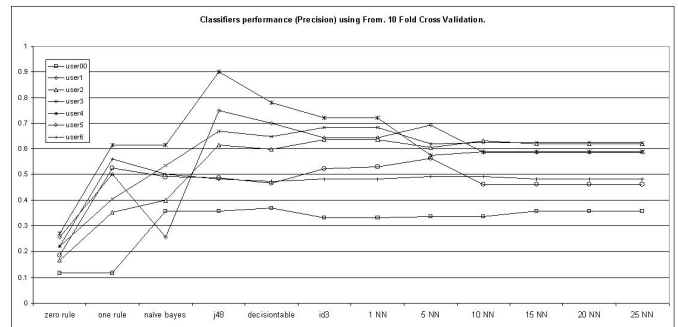


Fig. 8. Performance measured using precision of Classifiers on different users using 'From' only. 10 Fold Cross Validation.

The results also show the hypothesized bias that precision introduces when some class values are infrequent. User1 and user4 had high precision values compared to the other users and also had the most documents classified as uninteresting. Furthermore, this was the case when classification was done by zero rule, while zero rule produced the same recall across all users. See Figures 8, 9, 6 and 7. We thus believe that our measure was justified and achieves our objectives of reducing the effect of classification distribution and highlighting classifiers with good performance on important and interesting.

We found that accuracy was also biased heavily by the distributions of classifications. The differences between users

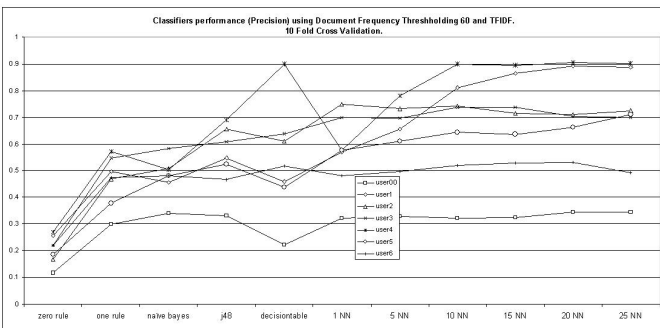


Fig. 6. Performance measured using precision of Classifiers on different users using Document Frequency Thresholding 60 and TFIDF. 10 Fold Cross Validation.

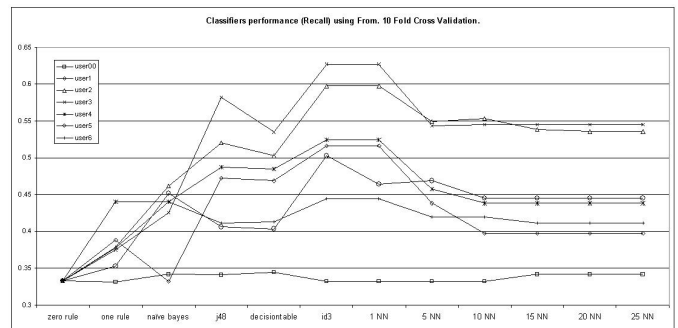


Fig. 9. Performance measured using recall of Classifiers on different users using 'From' only. 10 Fold Cross Validation.

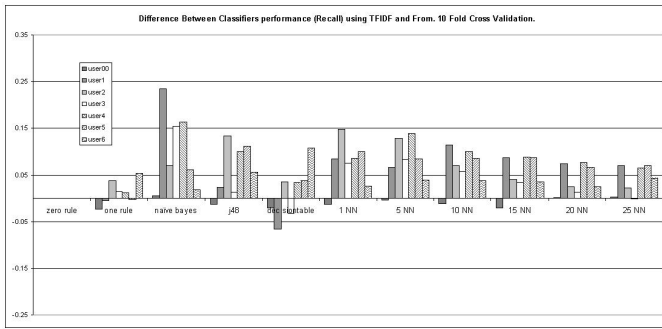


Fig. 10. Difference Between Recall of TFIDF and From on Classifiers. 10 Fold Cross Validation. Note that, the order that the learners appear on the x-axis of all these diagrams is the same. So, although the font is difficult to see a direct comparison can still be made.

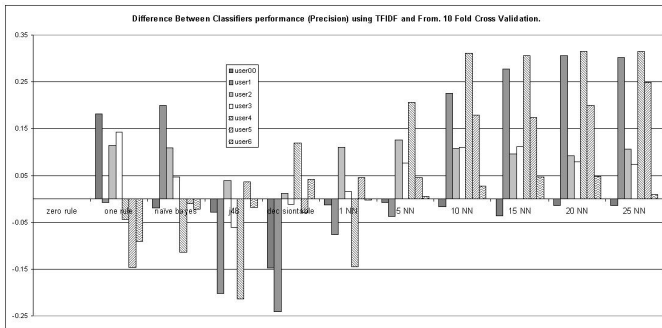


Fig. 11. Difference Between Precision of TFIDF and From on Classifiers. 10 Fold Cross Validation.

when measured with accuracy were highly correlated with the number of documents classified as uninteresting. Compare Figure 19 and Figure 4.

It was expected that TFIDF would perform better than From as it takes the body of the message into account and will be the focus of attempts to integrate background knowledge into the classifiers. This turned out to be the case, with performance using TFIDF significantly higher than for From over almost all classifiers and users. See Figure 10. The amount of benefit that was gained by using TFIDF varied between users and algorithms. When the performance was split up between recall and precision, some interesting trends emerged. The use of TFIDF increased recall in general and relatively uniformly across the classifiers and users, as can be seen in Figure 10. Precision was far less stable however. K nearest neighbors scored much higher in precision when using TFIDF for high values of K, while other algorithms, notably j48, scored lower in precision when using TFIDF. See Figure 11.

Since the dataset is made up of a large amount of mailing list items and announcements, which all generally come from the same source, the use of From would be expected to do well if there was little variation in the topic of the mailing list. For example, a mailing list for machine learning would most likely be always interesting to a user who works in this field and thus From would be an excellent indicator. On the other hand, announcements of talks or seminars vary in topic. Hence,

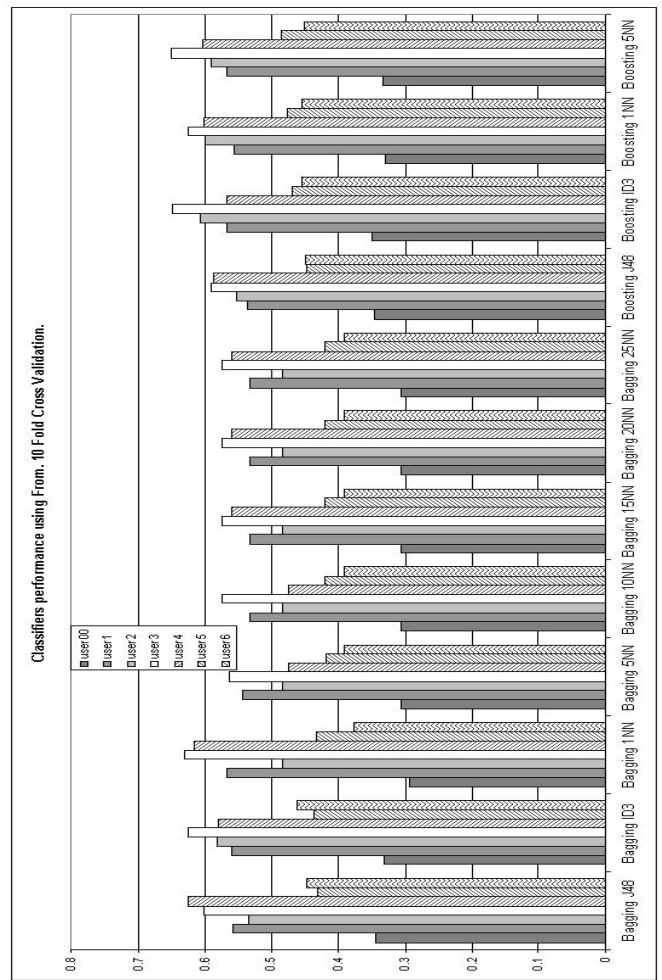


Fig. 12. Performance of Bagged and Boosted Classifiers on different users using 'From' only. 10 Fold Cross Validation. Bagging was done using 5 bags and 100% resampling.

unless a user does not care about the content, TFIDF would be expected to be more useful and From might be less useful. On the other hand, a message from a mailing list dedicated to one topic should also be detected well by TFIDF. Thus overall, TFIDF should perform better.

F. Effect of Bagging and Boosting

When using From as a feature, Bagging tended to either reduce the performance or leave it roughly unchanged. Boosting did not have much effect for all classifiers except 5 nearest neighbors, where the performance was increased for most users. See Figures 2 and 12. Note that we did not perform any boosting for K higher than 5.

For TFIDF, bagging increased the performance of j48 for all users, but did not significantly alter the other algorithms. Boosting j48 dramatically increased its performance for all users except the random user. Boosting of K neighbors showed no significant change. Refer to Figures 3 and 13.

This question is of importance for applications that assist users with management of their email. For this purpose we

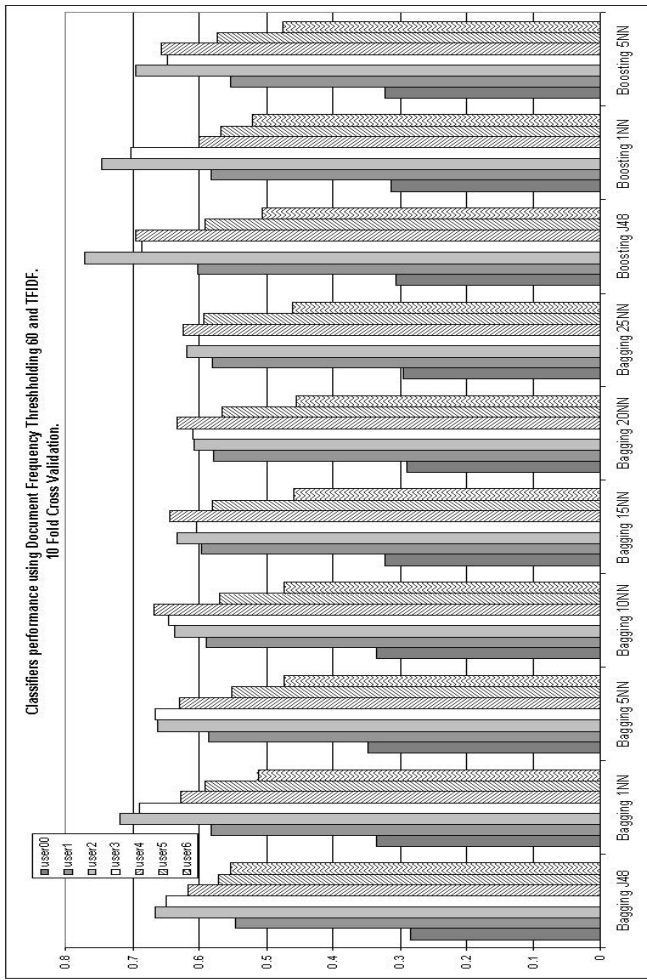


Fig. 13. Performance of Bagged and Boosted Classifiers on different users using Document Frequency Thresh-holding 60 and TFIDF. 10 Fold Cross Validation. Bagging was done using 5 bags and 100% resampling.

suggest that out of the reported measures, our measure is most useful in picking a classifier.

Our best performance was boosted J48 on user2 at 0.77, the second highest was 1 nearest neighbors on user2 at 0.75. These corresponded to an accuracy of 78% and 75% respectively. In terms of accuracy, 85% is easily achieved by many classifiers on user4, and bagging of 15, 20, and 25 nearest neighbors achieved over 90% for user4. Many other users hovered just under 80% accuracy. We believe that this is usable.

Overall, 1 Nearest Neighbors, ID3 and J48 seem to do best, and this is true for all users in general. Boosting J48 significantly increases its performance, and this performed best overall. Furthermore, we showed that our measure of performance was indeed a good choice for our data set and application. Hence, we only report results using our measure for the following cross experiments.

V. CROSS-USER EXPERIMENTS

We have also begun to explore the potential power of community membership. Some of the users in our experiments

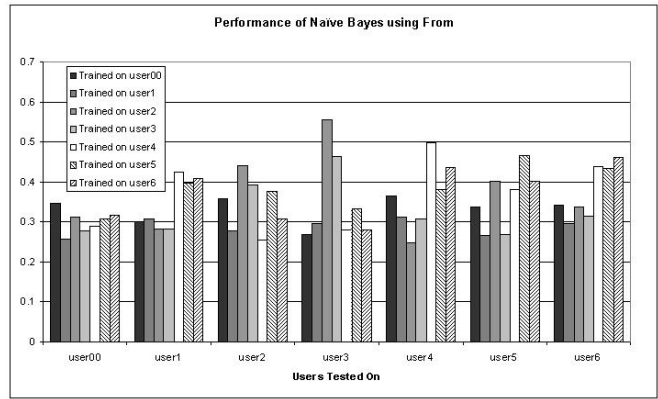


Fig. 14. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of Naive Bayes using 'From' only.

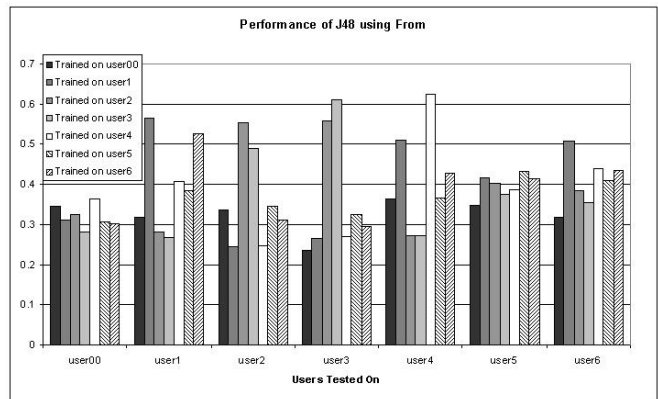


Fig. 15. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of J48 using 'From' only.

have very similar interests. This suggests that a classifier which was trained on one user might achieve good predictions of the classifications of another similar user. Although there are many ways to explore this, we chose a simple starting point. We train a classifier on one user, and then test it on all other users. (It is obviously impossible to use 10 fold cross validation for this!) Since we knew that some users in the group had similar interests, this evaluation gave the learner a real advantage: it would train on the same items for one user as it was tested on for other users.

The use of two separate features gives us an important way to interpret the results we obtained from these experiments. For example, if two users have a reflexive relationship expressed in the From results, this could mean that they have similar interests (an interpretation of the results), or simply that they are interested in data that comes from the same source (which is what the data is actually telling us). By observing the same patterns in the TFIDF results, it is more likely that the former holds. By reflexive we mean that the performance is high when trained on userX and tested on userY as well as when we train on userY and the test on userX.

A. Cross Experiments using From

We observed some interesting correlations between user1 and user4, user1 and user6, user 4 and user 6 and most prominently user2 and user3. The last relationship was especially interesting as both these people share many of the same research interests. Furthermore, users 1, 4 are both students.

Training on user2 and testing on user3 had consistently high values across all classifiers for From, see for example Figures 15, 16 and 17. The reverse, that is training on user3 and testing on user2 also had high values, although this tended to be slightly lower. This suggests that these two users classify their email similarly, which in turn implies that they have similar interests. In many cases, training on user2 and testing on user3 was almost as good as 10 fold cross validation on user3, as can be seen in Figure 18. In one case, notably when Naive Bayes was used, (Figure 14), training on user2 achieved higher performance than training on user3 when testing on user3. This suggests that any classifier produced from data classified by user2 would be very useful to user3.

Another similarity was evident between user1 and user4. Although not as profound, it was nevertheless prominent and consistent across most of the classifiers, as for example, in Figures 16 and 15. A further similarity arose between user 1 and user 6. This is especially prominent for J48 (Figure 15). Furthermore it seems to be reflexive and appears for other classifiers also. The question arises: if user1 is similar to both user4 and user6 (and vice versa), does this imply that user4 and user6 are also similar? That is, is this relationship transitive? The results seem to imply that there exists a small relationship. This is especially evident in J48 (Figure 15) but also appears in the other classifiers, notably 1 Nearest Neighbors (Figure 17 and ID3 (Figure 16).

K Nearest Neighbors : Variations in K did not seem to have any uniform or significant effect on performance. The largest change was between 1 and 5 nearest neighbors, the remaining results being very similar.

Effect of Classification Distribution : The similarity found between users as outlined above seems also to occur in their respective distributions. The following discussion refers to Figure 4. User1 and user4 have extremely similar distributions in classifications, while user2 and user3 have a similar number of items in the important category, although the number in the interesting category varies. It is unlikely that the distribution alone could account for the observed similarities, and in the case of user2 and user3, where the similarity is most prominent, the distributions vary more than for user1 and user4. User 1 and user 6 have quite different distributions. However, they appear to be similar. Hence we conclude that the cross experiments are sensitive to more than just the distribution and are in fact quite useful. However the possible effect of these distributions has to be kept in mind.

Effect of Bagging : Bagging using 66% sampling in general performed slightly worse than without bagging. This may well be due to only using two thirds of the data per bag and will be further investigated.

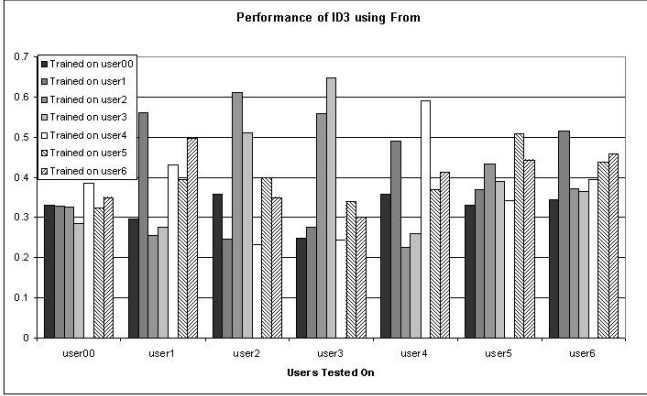


Fig. 16. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of ID3 using 'From' only.

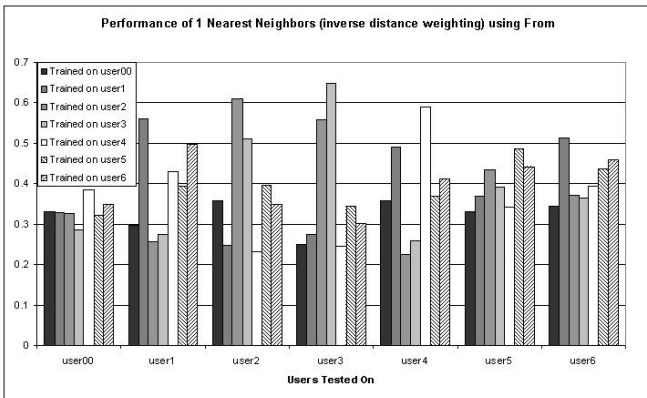


Fig. 17. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of 1 Nearest Neighbors using 'From' only.

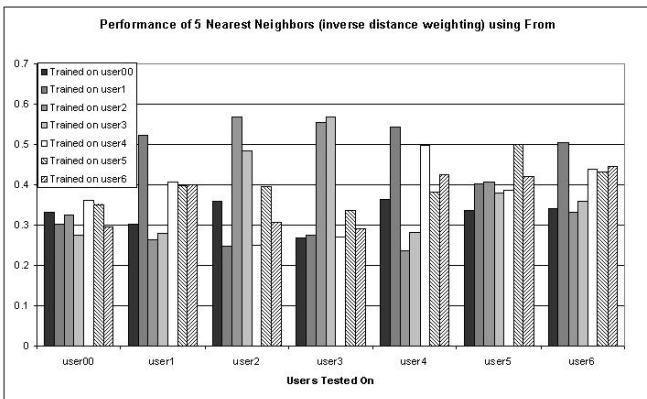


Fig. 18. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of 5 Nearest Neighbors using 'From' only.

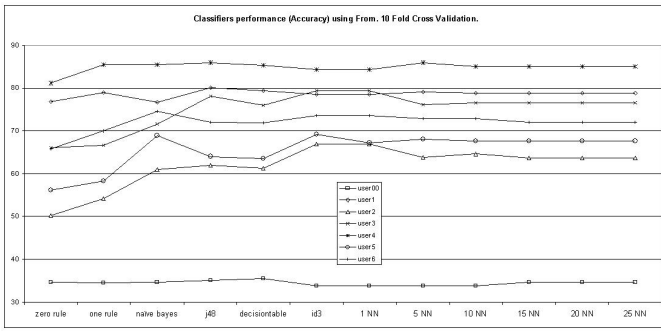


Fig. 19. Performance measured using accuracy of Classifiers on different users using 'From' only. 10 Fold Cross Validation.

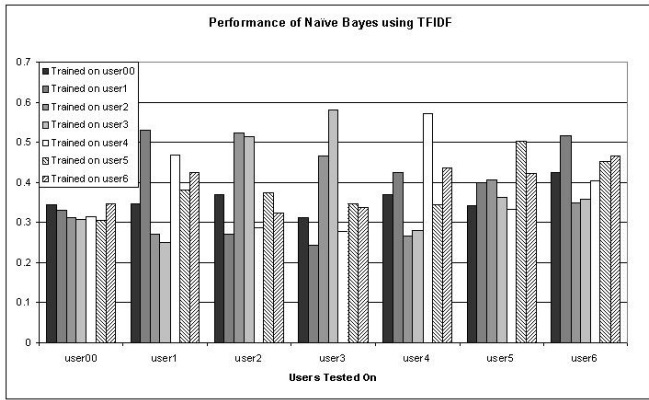


Fig. 20. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of Naive Bayes using 'Tfidf' only.

Overall, the same classifiers that worked well for individual experiments also exhibit the highest results for cross experiments. The similarities that occur in the results should be quite useful in the sharing of classifiers, at least for From. These similarities seem to be reflexive and somewhat transitive, which is very encouraging.

B. Cross Experiments using TFIDF

The same relationships that we observed using the From attribute also showed up using TFIDF. Namely, there was a strong and reflexive relationship between user2 and user3. Training on user3 and testing on user2 was consistently higher than the reverse over the classifiers. However, this margin was less than 0.01 generally. The second most prominent relationship was when we trained on user1 and tested on user6. This relationship was also somewhat reflexive but not to the extent of the relationship between users 2 and 3. Training on user1 and testing on user4 was the third most prominent relationship, and also tended to be reflexive, once again not as prominently as that between users two and three. User5 and user6 seemed to have a small relationship, while user4 and user6 did likewise. These relationships showed up quite uniformly over all the classifiers. Refer to Figures 20, 21 and 22.

It was very interesting to see that the patterns for From and TFIDF matched very closely. Comparing the graphs, one can

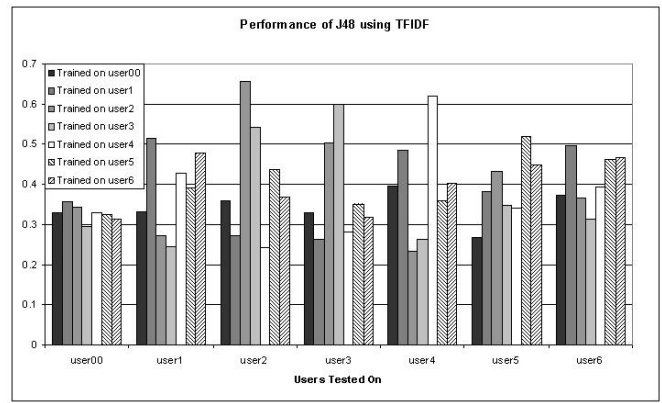


Fig. 21. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of J48 using 'Tfidf' only.

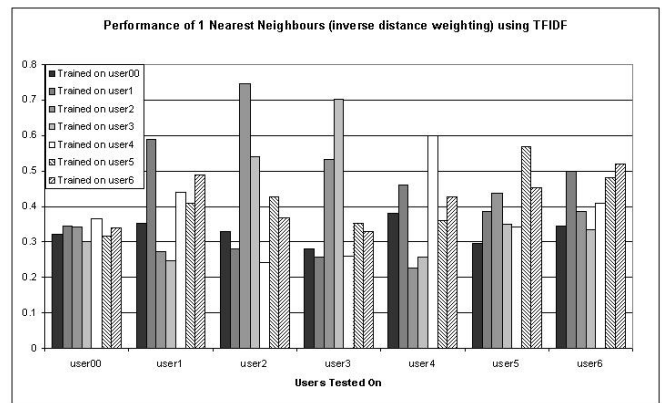


Fig. 22. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X == Y. Performance of 1 Nearest Neighbors using 'Tfidf' only. Note that, there is a small difference in the scale of the y-axis in comparison to the surrounding graphs.

see the same relative performances. This is very interesting as it means that From is a good predictor of performance on this data set. Furthermore, it shows that the similarities observed between users are likely to be significant. In general, the results using TFIDF were similar to those using From. It seems that TFIDF actually increases the similarities between the users that were identified. This effect is especially evident in K Nearest Neighbors. This was somewhat surprising as From is a much coarser feature we thought it was thus less likely to fully fit a users classification patterns and be better on others. Being able to predict similarities between users using the contents of the message at least as reliably as From is very encouraging.

K Nearest Neighbors : The performance between similar users increased as K increased for 1 to 5 (Figures 22 and 23). This does not reflect the same trend as was seen in the individual users, where the best performance was with low K. These results make sense as the data presented to the learner is far more noisy compared to the data it was trained on. And it is well known that increasing the value of K can smooth out the impact or noisy data points[12].

Effect of Bagging : Bagging using 66% sampling of the data

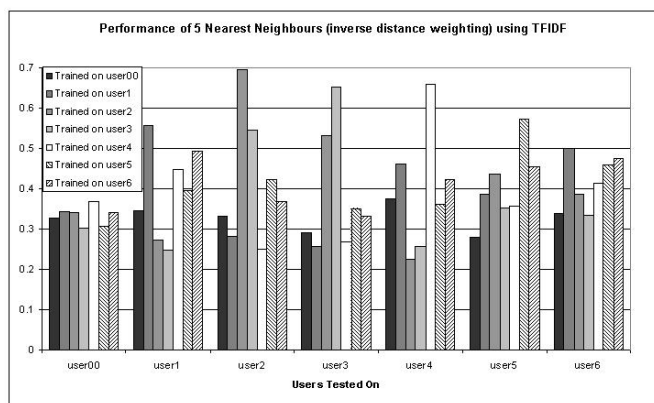


Fig. 23. Train on userX, test on userY for all X, Y. 10 fold cross validation used where X = Y. Performance of 5 Nearest Neighbors using "Tfidf" only.

on J48 did not seem to improve the results. The same technique on K nearest neighbors actually degraded the performance somewhat on average. We had expected bagging to improve the performance of classifiers in the cross experiments. This may have been due to using only 66% of the data when sampling, however.

Performance Measure : We did not evaluate cross experiments using measures other than two thirds recall, one third precision. This measure was quite similar to F1 and we have shown that our measure is appropriate.

J48 (Figure 21) and 5 Nearest Neighbors (Figure 23) were the best performers when using TFIDF for Cross user experiments.

VI. CONCLUSIONS

TFIDF was most successful in all types of experiments, but the results generally agreed well with From. Using classifiers to predict similarities between users by training on one persons data and testing on another's data was shown to be quite successful. The fact that the similarities were reflexive and somewhat transitive is very good news for applications in the sharing user models. The best classifiers were J48, ID3 and 1 Nearest Neighbor for From on individuals. Excluding ID3 which does not handle numeric attributes, these classifiers also did best when using TFIDF, although 1 Nearest Neighbor outperformed the others.

In the cross experiments, using a higher values of five for K was necessary for the best performance when using TFIDF, while K was generally not important when using From. The same classifiers that performed best for individual experiments also performed best for the cross experiments. This suggests that these classifiers are better for this problem domain generally, which is good news as it will allow us to focus our attention. Both decision trees and K Nearest Neighbor classifying algorithms are scrutible and K Nearest Neighbors lends itself especially well to visualization which may become very useful when applying these results to an application.

VII. SUMMARY AND OUTLOOK

At this stage in our research, we have succeeded in comparing classifiers for different users on our corpus, as well as comparing users to each other via a classifier. We have not yet incorporated background information into the classifiers, but we envisage doing this as a next step.

REFERENCES

- [1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. Spyropoulos. An experimental comparison of naive - bayesian and keyword-based anti-spam filtering with personal e-mail. In *23rd Int. ACM SIGIR Conf*, pages 160–167, 2000.
- [2] I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. Spyropoulos. 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*, pages 160–167, 2000.
- [3] C. Brutlag, J. Meek. Challenges of the email domain for text classification. In *Seventeenth International Conference on Machine Learning*, July 2000.
- [4] X. Carreras and L. Marquez. Boosting trees for anti-spam email filtering. In *4th Int. Conf. Recent Advances in NLP*, 2001.
- [5] J. Clark, I. Koprinska, and J. Poon. Linger - a smart personal assistant for e-mail classification. In *Proc. of the 13th Intern. Conference on Artificial Neural Networks (ICANN'03)*, pages 274–277, 2003.
- [6] 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.
- [7] E. Crawford, J. Kay, and E. McCreath. Automatic induction of rules for e-mail classification. In *In Proceedings of the Sixth Australasian Document Computing Symposium, Coff's Harbour, Australia*, 2001.
- [8] N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. *Interactions*, 8(5):30–38, 2001.
- [9] J Kay E Crawford and E McCreath. Iems - the intelligent email sorter. In *Proceedings of the ICML International Conference on Machine Learning*, pages 83 – 90, 2002.
- [10] H. Katirai. Filtering junk e-mail: A performance comparison between genetic programming & naive bayes, 1999.
- [11] E McCreath and J Kay. Iems : help users manage email. In *User Modeling - Lecture Notes in Computer Science*, volume 2702, pages 263–272, 2003.
- [12] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [13] 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.
- [14] J Provost. Naive-bayes vs. rule-learning in classification of email. Technical Report AI-TR-99-284, University of Texas at Austin, AI Lab, 1999.
- [15] J. Rennie. ifile: An application of machine learning to e-mail filtering. In *KDD-2000 Text Mining Workshop, Boston*, 2000.
- [16] 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.
- [17] G. Sakkis, I. Androutsopoulos, G. Paliouras, V.Karkaletsis, C. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. In *6th Conf. Empir.Meth. in NLP*, pages 44–50, 2001.
- [18] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 1999.
- [19] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US.