A Multi-Learner Approach to E-mail Classification

Elisabeth Crawford
School of Information Technologies
The University of Sydney
NSW 2002 Australia
ecrawfor@it.usyd.edu.au

Irena Koprinska
School of Information Technologies
The University of Sydney
NSW 2002 Australia
irena@it.usyd.edu.au

Jon Patrick
School of Information Technologies
The University of Sydney
NSW 2002 Australia
jonpat@it.usyd.edu.au

Abstract

The volume of e-mail which users receive has grown over the past few years. Most users try to cope with this problem by sorting e-mail into folders. In this paper we look at machine learning approaches to performing this classification automatically. In particular we describe a test and select approach to choosing both single learners and ensembles of learners to classify an individual user's e-mail. The results show that it is possible to select the most effective single learner for an individual user, but that selecting an ensemble without overfitting is more difficult. We also show that Widrow-Hoff is a highly effective algorithm for e-mail classification, and discuss the reasons for this.

Keywords Document Management, E-mail Management, Text Categorization, Machine Learning

1 Introduction

The management of personal e-mail is a significant and growing problem for individuals and organisations. Most e-mail users receive a wide variety of e-mail including spam, personal and business communications and e-mail associated with interest groups and mailing lists.

A study by Whittaker and Sidner [27] showed that e-mail is used for a wide variety of important tasks in offices. They found that it was used as a means of receiving and delegating tasks, for keeping track of tasks, and for ongoing correspondence. On average the users in this study received 49 new e-mails a day. Managing e-mail of this volume so it can be retrieved when necessary is a time consuming task.

Proceedings of the 7th Australasian Document Computing Symposium,
Sydney, Australia, December 16, 2002.

Not only do people receive a lot of e-mail associated with their work, most also receive a great deal of *spam* or *junk* e-mail. Junk e-mail has proliferated since the mid-1990s and it is an increasingly annoying and time wasting task for users to filter their e-mail. Cranor and LaMacchia [5] report that the "spam rate" per day in 1997 for the AOL system was approximately 30%.

E-mail users commonly try to manage their e-mail by sorting it into folders. Filing is a cognitively demanding task [11], thus it would be useful if it could be done automatically. Most commonly used e-mail managers such as Microsoft Outlook, allow the user to hand construct rules to automatically place e-mails containing particular keywords into specific folders. However, Duncheneaut and Bellotti [7] found that despite the potential usefulness of these rules most users do not create them. The majority of users in general avoided customising the software, many also find it difficult to figure out how to use software. In addition, manually constructing a set of robust rules is a very difficult task. Moreover, users are constantly creating, deleting and reorganising their folders. Even if the folders remain the same, the nature of the messages within the folder may well drift over time. The characteristics of the incoming junk e-mail (e.g. topics, frequent terms) also change over time. Hence the rules must be constantly tuned and refined by the user. This is a tedious and error-prone process. A system that can automatically *learn* how to classify e-mail messages into a set of folders is highly desirable. This is where Machine Learning (ML) can help. By studying how a user has classified their email in the past, ML algorithms can be used to build classifiers to associate incoming e-mails with folders.

Research in the area of Text Categorization (TC) has shown that it is possible to automatically build effective classifiers using ML techniques [23].

As a result several systems for automatic e-mail classification have been developed. Cohen [4] uses the propositional learning algorithm RIPPER to induce so called keyword-spotting rules. Bayesian approaches (Naive Bayes and more expressive Bayesian nets) were utilised in [14], [21], [20] and [1]. Nearest-neighbour techniques were used in [24], [12], [15]. In the domain of junk e-mail filtering, Provost [18] found that Naive Bayes outperforms RIPPER and Androutsopoulos et al. [1] reported that Naive Bayes compares favourably with a keyword-based filter similar to that used in Microsoft Outlook.

An important characteristic of e-mail data is the diversity of classification across users. Firstly, there is a lack of unique pre-defined categories; instead each user develops their own categories. Moreover, the type of these categories can vary greatly between users. For example, some users categorize their email according to how or when they need to deal with it; others categorize according to sender or topic. It is also not uncommon for the same user to create some categories according to topic, and others according to actions. This diversity poses a problem for determining what the best ML algorithm for email classification is. In fact there is unlikely to be one best algorithm for all users. This is supported by the results of Crawford, McCreath and Kay [6] who compared the following approaches on e-mail data of five different users: Naive Bayes, keyword rule learner similar to RIPPER, sender based rule learner, tf-idf based approach (see Section 3.2.1) and composite rule learner. It was found that the sender approach performed the best in terms of accuracy on two of the email sets, the keyword rule learner on another two, and the composite rule learner on one. Brutlag and Meek [2] also found that no one learner consistently outperformed the other.

In this paper we propose novel approaches to automatic classification of e-mails, that utilise a combination of ML algorithms. We explore the issues associated with managing the different types of folders created both by different users and by the same user to sort their e-mail. The first approach involves selecting the best single ML algorithm to use for a particular user. The second involves determining the best ensemble of ML algorithms for a particular user. We will first discuss our approach in more detail and then the evaluation of its effectiveness.

2 Multi-Learner Approach

Our approach is two-faceted. Firstly, the differences between the way in which users classify their e-mail has led us to explore an approach that attempts to determine the best ML algorithm to use for a given user. Secondly, as the same user often defines a number of different types of folders (categories), we wish to explore an ensemble of ML algorithms. Our aim is to determine if these two approaches can improve classification effectiveness and how they compare to each other.

In all our experiments we have used separate binary classifiers for each category. It is possible that a user may desire some e-mails to be stored in more than one folder, thus, we do not have to do any extra work (e.g. use error correcting codes) to combine the classifiers to solve the multi-class problem. Instead, when a new e-mail is presented for classification, each binary classifier determines whether or not it belongs in that category.

To determine the best ML algorithm and the best ensemble we propose a test and select method, similar to that described by Sharkey et al. [25] where a validation set is used to build an ensemble for fault diagnosis of a diesel engine. In the first case of our approach, we use the validation set to select the best performing learner overall, and in the second, we use it to select the best ensemble of learners, where a different learner may be chosen for each category.

There are two main differences between the approach of Sharkey et al. and ours. First, in their system the ensemble members are learners from the same type (only neural networks) and are generated using different initial conditions, architecture and data sampling. In contrast, we combine both modules and ensembles. Each module is responsible for one category and the ensembles are formed using one learner from each module. Second, in our approach the ensemble combines learners from different types.

Recently some other multi-learner approaches have been successfully applied for junk e-mail filtering. Sakkis et al. [22] combined a Naive Bayes and memory-based classifier by stacked generalisation and found that the ensemble achieved better performance. Several tree boosting methods were used in Carreras et al. [3] and it was shown that they compared favourably with decision trees, Naive Bayes and k-nearest neighbour. We study the potential of ensembles for general e-mail classification into several folders

2.1 Evaluating Classifier Effectiveness

In many applications of ML, the effectiveness of each algorithm is evaluated by the algorithm's *accuracy*. In TC however, the most commonly used evaluation metric is the *F-measure*. Below these two measures are defined in terms of the two-way contingency table for a binary classifier shown in Table 1.

E-mails	# from c_i	# not from c_i
# assigned to c_i	tp	fp
# not assigned to c_i	fn	tn

Table 1: Contingency Table (tp-true positives, tn-true negatives, fp-false positives, fn-false negatives)

Accuracy is defined as follows:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

In TC, however, each class typically has a small number of positive examples and a very large number of negative examples. It is thus possible for a classifier to have a very high accuracy value, without ever detecting a positive example of the class, as the *tn* term dominates the result. Hence, to get a clearer picture of how well a TC classifier has really learnt a class, the measures *precision* and *recall* are used:

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

To obtain an overall measure of effectiveness, the values of precision and recall are combined for the category using the *F-measure*:

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha)\frac{1}{R}}$$

where P stands for precision, R for recall and α determines their weighting. In some applications a high precision may be more important than a good value of recall or vice versa. In TC these terms are normally assigned an equal weighting and hence α is set at 0.5. When this is the case the metric is referred to as the *F1-measure* and it is this metric that we use in this paper.

To measure the average performance of a binary classifier over multiple categories, *micro-averaging* [23] is typically used. A summary two-way contingency table is formed by adding up the individual entries in the per-category tables. Then this table is used to compute the summary precision and recall measures.

By evaluating the micro averaged F-measure on the classifiers produced by each of a set of different ML algorithms, we can select for the user, the algorithm that we think will give the best performance on the basis of the validation set. Selecting the best ensemble, however, is more involved.

2.2 Forming the Ensemble

We want to choose the ensemble that when evaluated as a whole achieves the best micro-averaged F1-value. If we simply select the best learner for each category on the basis of the F1-value of the binary classifiers and combine them in an ensemble, there is no guarantee that this will be the best ensemble. Ensemble combinations work best when there is some diversity amongst their components, i.e. when they have different generalisation patterns and few coincident errors [25]. The following example shows that diversity among ensemble members might be more important than combing the best learners in an ensemble.

Suppose that we are selecting the ML algorithm for the ensemble and that all but one of the ML algorithms achieved a F1-measure of 0 on this folder. If we were selecting the best algorithm for the ensemble, we would select the ML algorithm with the positive F1-value. However, this may not be the best choice. Zero F1-values imply that the algorithm found no *true positives*. If an algorithm which found no true positives had very few *false negatives* (due to there being few positive examples of the class) and few *false positives* it may well add more to the ensemble than an algorithm that found only a few *true positives* at the expense of getting many *false negatives*.

What we are faced with is an optimisation problem, where we want to select the ML algorithm for each category that will maximise our objective function - the micro averaged F1-measure for the ensemble. For users who do not have a large number of different folders, we can easily find the ensemble that maximises the objective function using brute force. However, this approach is exponential in complexity; as the number of folders grows it becomes an unrealistic method to use. We can instead use a standard local search technique, such as hill climbing or tabu search to approximate the best solutions. We used a hill climbing approach, starting from an ensemble containing the best learner for each category.

3 Experimental Setup

3.1 Corpora Used for Evaluation

In 2001 Crawford et al. [6] collected a corpus of e-mail messages from five different users. The participants kept back any folders or e-mail messages they felt contained highly sensitive information.

Each of the users used different criteria to determine in what folder their e-mails belonged. For instance, user 1 categorized e-mails mostly on the basis of topic and sender information while user 4 also categorized messages according to when they needed to be acted upon. User 2 was different again, categorizing e-mails solely by the actions performed on them e.g Delete, ReplyAndKeep etc. Thus, the corpora is representative of a number of different styles of classification, despite its relatively small size (a problem common in all the studies of e-mail classification).

User	*Number of E-mails*	*Number of Folders*
1	526	7
2	430	9
3	869	12
4	972	39

Table 2: Statistics for the E-mail Corpora

To evaluate our approach we have used the first four corpora. They range form 430 to 972 e-mails in size and have between 7 and 39 categories, see Table 2. Crawford, Kay and McCreath [6] found that the User 2 and User 4 corpora were the hardest to automatically classify, which is in part explained by their relatively large category/number of e-mails, ratio.

3.2 Preprocessing of the Corpora

3.2.1 Text Representation

The first step in the process of constructing an automatic classifier is the transformation of the e-mails into a format suitable for ML algorithms. In TC this typically involves representing a document as a vector of weighted terms. The most common approach - the *bag of words* approach - uses words as terms with non-binary weights.

In the bag of words approach, all the unique words in the entire training corpus are identified. Each document is then represented by a vector that contains a weighting for every word in the corpus, which represents the importance of that word in the document. Methods for assigning weights to the different words vary, but the most popular is the normalised *term frequency, inverse document frequency - tfidf* approach. We have used the following tfidf function from [23]:

$$tfidf(t_k, d_j) = \#(t_k, d_j) * log\frac{|Tr|}{\#Tr(t_k)}$$

where $\#(t_k, d_j)$ is the number of times the term t_k occurs in the document d_j, and $\#Tr(t_k)$ is the *document frequency* of the term t_k, which is simply the number of documents in the training set Tr in which t_k occurs.

The tfidf values are usually normalised to fall inside the interval $[0, 1]$. We have used the cosine normalisation formula given in [23]. After applying stemming using the Porter Stemming algorithm [17] and removing stop words, this method of representing the text resulted in feature vectors of dimensionality between 4500 and 8600 for each of the corpora. This level of dimensionality was tractable for our machine learners and preliminary experimentation showed no consistent improvements in accuracy from reducing the number of features, therefore we have used the full feature set in our experiments.

3.3 Training, Validation and Testing Sets

To test our approach we have split the e-mails in each corpora into a training set, a validation set and a test set. The e-mails were ordered by the date the user received them and the first 40% were placed in the training set, the second 40% in the validation set and the final 20% in the test set. We created binary classifiers for each category and trained each of the ML algorithms described in the next section on the training set. We then tested each algorithm on the validation set, and used the results of these tests to select the best single learner for the user and the best ensemble according to the micro averaged F1 measure. Finally, we retrained each classifier on the combined training and validation set, and tested the performance of each learner and our ensemble on the test set. In accordance with the standard procedure in TC research, we only train classifiers for folders that contain at least one positive test example (note that this eliminates some folders).

3.4 Machine Learning Algorithms

We have used six different ML techniques to build the email classifiers. We have tried to select a large variety of learners, so that our ensembles may be as rich as possible. We have also been careful to include SVMs and Widrow-Hoff as these algorithms are two of the best performers in TC [23]. The purpose of this is two-fold. Firstly, it means that these highly effective algorithms can be included in our ensembles, and secondly, it ensures that we are comparing our ensemble approach with state of the art classifiers. Below is a brief description of the ML algorithms we have used and their parameters.

- *Support Vector Machines*
  Support Vector Machines (SVMs) [26] are a recently developed ML technique. They find the maximum margin hyperplane between two classes using the the training data and applying an optimisation technique. SVMs have shown good generalisation performance on many classification problems, including TC [8]. What makes them particularly suitable for TC is their ability to handle high dimensional features and their automatic tuning of parameters [10]. We have used the Weka implementation [28] of the SMO algorithm for SVMs [16] with both linear and quadratic kernels.

- *K-Nearest Neighbour*
  K-Nearest Neighbour (KNN) is the most commonly used memory based ML algorithm. It is a lazy learner as it stores all the training samples and does not build a classifier until

a new example is presented for classification. A distance metric is used to find the similarity between the new example and all training examples. The class is then decided by the majority vote of the closest k instances. In our experiments we used the Weka [28] implementation of the kNN algorithm, with cosine similarity as the distance metric and distance weighted voting, for k equal to one, five and thirty.

- *Decision Tree*
  Decision trees (DTs) are the most popular inductive learning algorithm. The nodes of the tree correspond to attribute tests, the links - to attribute values and the leaves - to the classes. To induce a DT, the most important attribute (according to the information gain) is selected and placed at the root; one branch is made for each possible attribute value. This divides the examples into subsets, one for each possible attribute value. The process is repeated recursively for each subset until all instances at a node have the same classification, in which case a leaf is created. To classify an example we start at the root of the tree and follow the path corresponding to the example's values until a leaf node is reached and the classification is obtained. To prevent overtraining DTs are typically pruned. The most popular DT learning package is Quinlan's C4.5 [19]. We have used the Weka implementation [28] of the C4.5 algorithm in our experiments.

- *Perceptron*
  A perceptron [9] is a simple type of neural network. It tries to induce a linear decision boundary that separates the positive and negative examples in the training data. The learning process involves a number of passes over the entire training set. After the presentation of each example, the boundary is adapted (moved towards or away from the example) depending on how the example was classified (correctly or not). The algorithm terminates when all the training examples are correctly classified or when the epoch limit is reached. A new example is classified by determining which side of the boundary it lies on. The main disadvantage of the perceptrons is that they can only separate linearly separable data and are not able to produce a theoretically good boundary, e.g the one that defines the maximum separation between positive and negative examples. However, perceptrons are quite fast and have been very effective in the TC domain. We have used the Weka implementation [28] of the perceptron algorithm in our experiments.

- *Widrow-Hoff*
  The Widroff-Hoff (WH) algorithm [9] is a neural network algorithm that learns a linear decision boundary similarly to the perceptrons. It adjusts the weights of the network in order to minimise the mean squared error of the difference between the actual and target outputs of the network for each training example. This amounts to an approximate steepest gradient descent on the error surface. Widrow-Hoff can not learn non-linear decision boundaries, however if run on non-linear data it will try to find the linear boundary that results in the lowest mean squared error. This makes it more noise tolerant and allows for better generalisation. We have used our own implementation of the standard Widrow-Hoff algorithm in our experiments.

- *Naive Bayes*
  Naive Bayes (NB) is a simple but highly effective Bayesian learner. It uses the training data to estimate the probability that an example belongs to a particular class. It assumes that attribute values are independent given the class. This assumption clearly has no basis in many learning situations including TC, nonetheless Naive Bayes can produce very good results. For more information on Naive Bayes see [13]. We have used the Weka implementation [28] in our experiments.

4 Results and Discussion

4.1 Determining the Best ML Algorithm for Each User

Table 3 presents the results for each user on validation and test set. Each entry represents the micro averaged (over all folders) F1 measure and the best results are given in italics. A comparison between the results on the validation and test set for the different learners indicates similar performance patterns. Hence the validation set can be effectively used to select the best single learner for each user. Once the best learner for each user is selected using such a test and select approach, the classifier can be retrained on all e-mails sorted by the user (i.e. training, validation and test set) and used for future automatic classification.

Contrary to [6] and [2], we found that the same learner performed the best for all the users. WH, which to the best of our knowledge has not been applied to e-mail classification before, performed better than any other single learner on each corpus. Linear SVMs (SVM1) also performed well, coming second for users 1 and 3 and third for user 4. Another linear classifier - perceptron - showed relatively good performance, while nearest neigh-

Learner	User1 Validation	User1 Test	User2 Validation	User2 Test	User3 Validation	User3 Test	User4 Validation	User4 Test
KNN1	0.157	0.506	0.149	0.406	0.642	0.73	0.563	0.34
KNN5	0.607	0.719	0.056	0.247	0.498	0.596	0.568	0.484
KNN30	0.65	0.715	0.0	0.475	0.281	0.468	0.366	0.364
SVM1	0.668	0.794	0.377	0.406	0.807	0.807	0.551	0.585
SVM2	0.662	0.758	0.161	0.379	0.603	0.756	0.428	0.516
C.45	0.621	0.713	0.362	0.434	0.769	0.782	0.528	0.573
NB	0.614	0.697	0.365	0.387	0.624	0.471	0.265	0.241
Perceptron	0.667	0.72	0.373	0.397	0.562	0.773	0.329	0.593
WH	*0.688*	*0.806*	*0.434*	*0.485*	*0.838*	*0.824*	*0.589*	*0.609*

Table 3: Single Learner Results for Each User

bour was inconsistent and Naive Bayes performed the worst of all.

It is worth considering why our results do not support the previous results of Crawford et al. [6], indicating that no one learner consistently outperformed the others. We cannot directly compare the results due to our different experimental setup, however the answer probably lies in the difference between the learners being evaluated. Unlike the learners used in [6] WH and SVMs are two of the most effective TC methods. Furthermore, the strong performance of WH, linear SVM and perceptron imply that sophisticated linear methods work well for email classification. Crawford et al. [6] have largely used inductive rule based approaches; the only linear approach they use is NB which was the worst performer in our experiments. The results strongly suggest that WH is a more consistently good performer across users than previous approaches.

The results shown in Table 3 demonstrate a strong difference in the difficulty of automatically categorizing different users' e-mails. Consistent with [6] our results show that the coarser grained e-mail sets of users 1 and 3, where many e-mails were classified according to topic, were easier to classify than the finer grained user 2 and user 4 corpora. The fact that these corpora were more finely grained meant that for each binary classifier there was less training data, making the classification harder to learn. Further, these second e-mail sets were also classified more according to actions, which intuitively seems like a harder sort of classification to learn with TC methods.

With the exception of Naive Bayes our results indicate that holistic linear algorithms (WH, SVM1, perceptron) perform relatively well compared to non-linear approaches (KNN1, KNN5, KNN30) and inductive keyword based ones (C4.5). The linear models have a strong advantage over models like DTs in that not only can they recognise key features (e.g. in WH connection weights indicate the importance of features), but they can also make an assessment using all available information which is important given the amount of concept drift found in e-mail.

The very poor performance of Naive Bayes can be attributed to overzealous predictions that e-mails belong to the class in question. The Naive Bayes classifiers all had very high recall, but at the expense of very low precision due to many false positives. This problem could perhaps be lessened by thresholding the predictions, so that the classifier cannot predict membership unless the probability of membership is significantly higher than the probability of non membership.

K-nearest neighbour is a simple method which was found to perform best on a TC task(Reuters) when compared to 13 other methods [29]. However, we found that it was inconsistent and often performed poorly on e-mail classification. Nearest neighbour approaches perform well when the data is not noisy and when each attribute contributes approximately equally to the classification. Compared to news wire texts (e.g Reuters), e-mail contains a great deal of noise, due to the vastly different writing styles of e-mail senders and most probably inconsistencies in classifications. Further, the lack of a generalised model in a domain where certain key-words really do matter more than others (e.g. the sender's name is an important feature) is a serious disadvantage.

Finally, it is worth considering why WH performs better than the perceptron and SVM classifiers. WH classifiers are more powerful than perceptrons as they find a boundary which minimises the classification error while perceptrons simply look for a boundary that works. The weaker performance of the SVMs with the quadratic kernel (SVM2) versus the linear kernel simply implies that a linear model more effectively represents that data. It is surprising however, that WH performs better than the linear kernel SVM and this issue requires further investigation.

4.2 The Evaluation of the Ensemble

The results are summarised in Table 4. It shows the ensemble members selected for each folder based on

User	Ensemble	Validation	Test
User1	[WH,SVM1,SVM2,WH]	0.688	0.8
User2	[C4.5,KNN30,SVM1,SVM1,SVM2,WH]	0.473	0.504
User3	[WH,KNN5,WH,SVM1,C4.5,SVM2,WH,SVM1,SVM1,WH]	0.858	0.788
User4	[WH,WH,KNN5,WH,KNN1,KNN1,WH,NB,WH,WH,WH,WH,SVM1,WH]	0.625	0.524

Table 4: Ensembles and F1 Results on Validation and Test Set for Each User

their performance on the validation set, together with the overall micro averaged F1 measure for the respective ensemble on the validation and test set. By comparing the results on the test set in Tables 3 and 4, it can be seen that the ensemble outperformed the best single learner (WH) on user 2's e-mail, obtained almost the same performance on user 1 and was less successful on users 3 and 4.

The failure of the approach on three of the corpora has more to do with the method of ensemble selection, than with a problem inherent with the idea. It is possible to choose on the test set an ensemble that is at least as good as the best single learner (as the ensemble members can all be from the same learner), however such ensembles are not being chosen on the basis of the validation set. By comparing the performance on the validation sets for users 2, 3 and 4, it can be seen that the ensembles (Table 4) performed considerably better than the best single learners (Table 3). The issue is then to explain, why the best ensemble for the validation set is not the best ensemble for the test set and how we can go about selecting a better ensemble.

Clearly we are overfitting the validation set when we choose the ensemble. This problem is most likely compounded by concept drift and the problem of sampling. During the period of time covered by the validation set, the user for example may have received a lot of e-mails about certain topics and the ensemble selected reflects this. However, during the period covered by the test set, the user may not have received many e-mails on these topics at all. This could either be due to true concept drift (the user may never receive many e-mails on this topic again redefining the overall nature of the folders) or it may be a problem with sampling - perhaps the user will receive more e-mails on these topics in a few weeks time. It is most likely that both of the above contributed to the overfitting. This is reinforced by the fact that user 2's email was collected over a two week period only, whereas the other users' e-mails range over periods from 1 to 5 months. Thus it is likely that the approach worked well for user 2 as there was much less room for concept drift and sample induced differences.

We have done a small amount of experimentation with different training/validation/testing splits that has suggested that some members of the ensemble are more stable than others (i.e. some were regularly selected for particular categories). This indicates the need for an ensemble selection process that is more resistant to sampling problems.

5 Conclusions and Future Directions

This paper has shown that a test and select method can effectively be used to choose the most effective learner for building a classifier for an individual's email. On each of our four test corpora the Widrow-Hoff algorithm achieved the best performance in terms of micro averaged F1. Neural network classifiers are often thought of as being inscrutable, however simple neural classifiers such as Widrow-Hoff can be made understandable for email users. Using rule extraction techniques from trained neural networks [30], we can communicate to the user the reasons for classification decisions in the same way we can when using rule based techniques [6] [4]. Thus we can take advantage of the superior performance of the Widrow Hoff algorithm without sacrificing scrutability.

We have also explored a test and select approach for forming an ensemble that uses different learners for different folders. This approach was successful in improving classification performance on one of the four corpora. We believe ensembles with better generalisation could be selected for the other three corpora using more sophisticated methods. To determine the ensemble it would be worth exploring the use of the sliding window tester described in [6] as this approach should be more resistant to sampling problems. We would also like to collect datasets of e-mails that span a number of months from users that receive a large amount of email, so that we can test our ideas on more corpora. Finally, the work in this paper further confirms the need for benchmark e-mail corpora allowing consistent evaluation of the approaches for automatic e-mail classification.

6 Acknowledgements

The first author would like to thank the Capital Markets CRC, CMCRC Ltd. for their support.

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 messages. In *23rd Int. ACM SIGIR Conf. pp. 160-167*, 2000.

[2] J. Brutlag and C. Meek. Challenges of the email domain for text classification. In *17th Int. Conf. on Machine Learning*, 2000.

[3] X. Carreras and L. Marquez. Boosting trees for anti-spam email filtering. In *4th Int. Conf. on Recent Advances in NLP*, 2001.

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

[5] L. Cranor and B. LaMacchia. Spam! *Comm. of the ACM v.41 n.9 pp.74-83*, 1998.

[6] E. Crawford, J. Kay and E. McCreath. Iems - the intelligent email sorter. In *19th Int. Conf. on Machine Learning, ICML*, 2002.

[7] N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. *Interactions v.8, n.5, pp.30-38*, 2001.

[8] S. Dumais, J. Platt, D. Heckerman and M. Sahami. Inductive learning algorithms and representations for text categorization, 1998.

[9] J. Hertz, A. Krogh and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[10] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *10th European Conf. on Machine Learning*, 1998.

[11] A. Kidd. The marks are on the knowledge worker. In *CHI'94 Conf. on Human Factors in Computing Systems pp.186-191*, 1994.

[12] P. Maes. Agents that reduce work and information overload. *Comm. of the ACM, v.37, n.7, pp.31-40*, 1994.

[13] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

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

[15] T. Payne and P. Edwards. Interface agents that learn: An investigation of learning issu es in a mail agent interface. *Applied Artificial Intelligence, v.11, p.1-32*, 1997.

[16] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, 1998.

[17] M. Porter. An algorithm for suffix stringing. *Program, v.14, n.3, pp. 130-137*, 1980.

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

[19] J.R. Quinlan. *C4.5: Programms for Machine Learning.* Morgan Kaufmann, 1993.

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

[21] 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.

[22] 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. on Empirical Methods in NLP, pp.44-50*, 2001.

[23] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys, v.34, n.1, pp.1-47*, 2002.

[24] R. Segal and M. Kephart. Mailcat: An intelligent assistant for organizing e-mail. In *3d Int. Conf. on Autonomous Agents, pp.276-282*, 1999.

[25] A. Sharkey and E. Sharkey. The "test and select" approach to ensemble combination. In *First Int. Workshop on Multiple Classifier Systems.* LNCS, 2000.

[26] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[27] S. Whittaker and C. Sidner. Email overload: Exploring personal information management of email. In *CHI, pp. 276-283*, 1996.

[28] I. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implm.* Morgan Kaufmann, 2000.

[29] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval v.1, n.1/2, pp.69-90*, 1999.

[30] Z. Zhou, Y. Jiang and S. Chen. Extracting symbolic rules from trained neural network ensembles. *AI Comm., in press*, 2003.