AN ABSTRACT OF THE THESIS OF

Victoria L. Keiser for the degree of Master of Science in Computer Science presented on May 22, 2009.
Title: Evaluating Online Text Classification Algorithms for Email Prediction in TaskTracer

Abstract approved:

Thomas G. Dietterich

This paper examines how six online multiclass text classification algorithms perform in the domain of email tagging within the TaskTracer system. TaskTracer is a project-oriented user interface for the desktop knowledge worker. TaskTracer attempts to tag all documents, web pages, and email messages with the projects to which they are relevant. In previous work, we deployed an SVM email classifier to tag email messages. However, the SVM is a batch algorithm whose training time scales quadratically with the number of examples. The goal of the study reported in this paper was to select an *online* learning algorithm to replace this SVM classifier. We investigated Bernoulli Naïve Bayes, Multinomial Naïve Bayes, Transformed Weight-Normalized Complement Naïve Bayes, Term Frequency – Inverse Document Frequency counts, Online Passive Aggressive algorithms, and Linear Confidence Weighted classifiers. These methods were evaluated for their online accuracy, their sensitivity to the number and frequency of classes, and their tendency to make repeated errors. The Confidence Weighted Classifier and Bernoulli Naïve Bayes were found to perform the best. They behaved more stably than the other algorithms when handling the imbalanced classes and sparse features of email data.

Evaluating Online Text Classification Algorithms for Email Prediction in TaskTracer

by
Victoria L. Keiser

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 22, 2009
Commencement June 2009

Master of Science thesis of Victoria L. Keiser presented on May 22 2009.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering & Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Victoria L. Keiser, Author

ACKNOWLEDGMENTS

I would like to acknowledge my major advisor, Tom Dietterich, for his guidance in performing my research, for providing the data to test, and for his assistance in writing this paper.

I would also like to acknowledge the past and present TaskTracer team for laying the groundwork which made this research possible and for their helpful suggestions.

Finally I would like to thank my husband for his support and his patience with many late nights.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA, or the Air Force Research Laboratory (AFRL). This work was also partially funded by a gift from Intel Corporation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

Evaluating Online Text Classification Algorithms for Email Prediction in TaskTracer

1. INTRODUCTION

The TaskTracer system [5] is an intelligent activity management system that helps knowledge workers manage their work based on two assumptions: (a) the user's work can be organized as a set of ongoing activities such as “Write TaskTracer Paper” or “CS534 Class”, (b) each activity is associated with a set of resources. “Resource” is an umbrella term for documents, folders, email messages, email contacts, web pages and so on. The key function of TaskTracer is to tag resources according to the activities to which they are relevant. Once resources are tagged, TaskTracer can help the knowledge worker recover from interruptions, re-find previously-visited resources, and triage incoming email.

Most resources, including documents, web pages, and file system folders, are tagged at the time they are visited by the user based on the “current project” of the user. However, because email arrives asynchronously, it requires a different approach. In previous work [13], we developed and deployed a hybrid learning system that classifies email messages as they arrive. This classifier employs a standard SVM classifier (based on libSVM) to make classification decisions. A companion Bernoulli Naive Bayes classifier provides a confidence estimate, which is employed to decide whether to use the SVM’s prediction. The hybrid classifier is fully integrated into Microsoft Outlook via a VSTO Addin. It provides a convenient user interface for providing and correcting email tags.

While our hybrid classifier is reasonably accurate, it is quite slow. The SVM is trained via the standard batch (SMO) algorithm, which scales approximately quadratically with the number of examples. Hence, as more and more email arrives, the classifier requires unacceptably large amounts of time to train. In addition, batch training requires storing all of the training examples, which is undesirable for both practical and policy reasons.

The goal of this research was to compare six state-of-the-art *online* classifiers to determine which would be best to deploy within TaskTracer. In our work, we have made the following assumptions:

- Email messages are associated with exactly one activity (class). Although there are multilabel document classification methods, they are not nearly as mature as standard multi-class classifiers.
- There are hundreds of classes.
- The classifier must be trained online in time linear in the size of the email message and linear in the number of classes.
- The set of classes changes over time as different activities rise and fall in importance.

2. ALGORITHMS

Six different text classification algorithms were examined: Bernoulli Naïve Bayes, Multinomial Naïve Bayes, Transformed Weight-Normalized Complement Naïve Bayes, Term Frequency-Inverse Document Frequency Counts, Online Passive Aggressive, and Confidence Weighted.

Bernoulli Naïve Bayes (BNB) is the standard Naïve Bayes classification algorithm which is frequently used in simple text classification [11]. BNB estimates for each class c and each word w, $P(w \mid c)$ and $P(c)$, where w is 1 if the word appears in the document and 0 otherwise. A document is predicted to belong to the class c that maximizes $P(c)\ \Pi_w\ P(w \mid c)$, where the product is taken over all words in the lexicon. (This can be implemented in time proportional to the number of words present in the document.)

Multinomial Naïve Bayes (MNB) is a variation on Bernoulli Naïve Bayes [3] in which w is a multinomial random variable that indexes the words in the lexicon, so $P(w|c)$ is a multinomial distribution. We can conceive of this as a die with one "face" for each word. A document is generated by first choosing the class according to $P(c)$ and then rolling the die for class c once to generate each word in the document. A document is predicted to belong to the class c that maximizes $P(c) \Pi_w P(w \mid c)$, but now the product w is over all appearances of a word in the document. Hence, multiple occurrences are captured.

Rennie et al. introduced the Transformed Weight-Normalized Complement Naïve Bayes (TWCNB) algorithm [10]. This improves MNB through several small adaptations. It transforms the feature count to pull down higher counts while maintaining an identity transform on 0 and 1 counts. It uses inverse document frequency to give less weight to words common among several different classes. It normalizes word counts so that long documents do not receive too much additional weight for repeat occurrences. Instead of looking for a good match of the target email to a class, TWCNB looks for a poor match to the class's complement. It also normalizes the weights.

Term Frequency-Inverse Document Frequency (TFIDF) is a set of simple counts that reflect how closely a target email message matches a class by dividing the frequency of a feature within a class by the log of the number of times the feature appears in messages belonging to all other classes. A document is predicted to belong to the class that gives the highest sum of TFIDF counts [3].

Crammer et al [3] introduced the Online Passive Aggressive Classifier (PA), the multiclass version of which uses TFIDF counts along with a shared set of learned weights. When an email message is correctly predicted by a large enough margin, the weights are not changed ("passive"). When a message is incorrectly predicted, the weights are aggressively updated so that the correct class would have been predicted by a margin of 1.

Confidence Weighted Linear Classification (CW) is an online algorithm introduced in Dredze et al [6]. It also makes use of a weight vector, which is updated more aggressively for parameters in which the classifier has less confidence and less aggressively for parameters in which it has more confidence.

3. PREVIOUS RESEARCH

Many researchers have studied email classification, particularly in the context of email foldering. Bekkerman et al. [1] compared a variety of algorithms (including MNB and SVMs) on the Enron and SRI/CALO email corpora. Of these, only MNB was a fully online algorithm. They found that SVMs performed the best, with logistic regression (maxent) second, wide-margin Winnow third, and MNB worst. Many authors have studied the performance of Naïve Bayes, SVMs, and Ripper on email classification tasks including Cohen [2], Provost [8], Rennie [9], Kiritchenko and Matwin [6], and Segal and Kephart [12]. An open source tool, POPfile (popfile.org) provides an implementation of Naïve Bayes that integrates well with POP email clients. To our knowledge, no previous work has compared the broad set of online algorithms evaluated in this paper.

Dietterich et al [4] also investigated user email behaviors with tasks. They track several Intel knowledge workers, finding them to work on an average 16 tasks per month and 7 tasks per day and to open an average approximately 17 emails per day.

4. DATA SET

The data set consists of email received and tagged (using the TaskTracer user interface) by the author’s major advisor. Previous research has shown that email folder structures can be difficult to predict, because their semantics can vary greatly. One potential advantage of TaskTracer tags is that they correspond to on-going activities which are likely to be more stable and more predictable than arbitrary user-defined folders. Spam has already been filtered from the email set, leaving almost 21,000 examples, dating from 2004 to 2008. There are 380 classes, ranging in size from a single message to 2500 messages. Comparing the user which provides our data set to those studied by Dietterich et al [4], he is atypical in his reading five times more emails per day and working on ten times more tasks per month. However, he works on a typical number of tasks per day, despite the larger overall volume. While not entirely typical, this data set should still be meaningful with the large number of labeled email examples over an extended period of time.

Feature extraction is performed as follows. One boolean feature is defined for each unique sender/recipient email address. In addition, one boolean feature is defined for each unique set of recipients (including the sender) as a proxy for the "project team". One boolean feature is defined for each unique word in the email subject and body with the exception of stopwords. No stemming is performed. There were a total of 84,247 features. We will refer to this as the Full data set. Many tests are run on a smaller feature set, which does not include words in the body. This yields 21,827 features, and we will call it the NoBody data set. The data sets are otherwise the same, based on the same email messages.

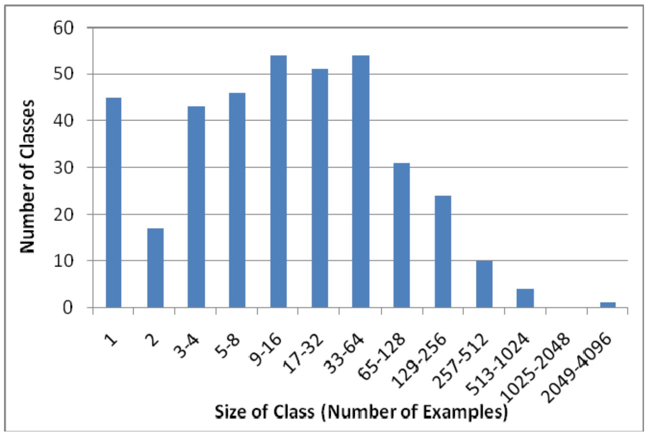

Figure 1 – Histogram of frequency of class size.

5. EXPERIMENTAL PROTOCOL

We ran each of the different algorithms on the Full and NoBody data sets. We follow the standard online learning protocol: Messages are processed in the order they were received. Each message is first predicted by the algorithm and that prediction is then scored as correct/incorrect. (We also retain the confidence of each prediction so that we can produce precision/coverage curves.) After prediction, the message with its correct class label is given to the online algorithm to update the classifier.

We compute several measures of performance. First, we plot the precision versus the coverage. This is performed by varying a confidence threshold and scoring the precision of those predictions where the confidence was above the threshold. This is computed for the entire period covered by the data sets. Second, we plot the cumulative error rate under 100% coverage. Third, we performed a series of analyses to understand how accuracy relates to the number of training examples in each class. Finally, we have noticed in our current system that the SVM classifier tends to make "the same" error repeatedly. This is very annoying for the user, who finds him/herself repeatedly correcting the learning algorithm and wondering when it is going to figure out its mistakes. Two of the learning algorithms, PA and CW, try to make big corrections when an error is made so that error will not be immediately repeated. We wanted to see whether these corrections achieve this effect.

6. RESULTS

Figures 2 and 3 show the tradeoff between precision and coverage for each algorithm. On the Full dataset, CW attains the highest accuracy across the range of coverage. On the NoBody dataset, BNB performs slightly better than CW at 100% coverage, but CW is still better at lower coverage levels. MNB performs badly until coverage is very low. TWCNB, PA, and TFIDF all perform very similarly, each performing best among the three at different times, but never outperforming either CW or BNB.

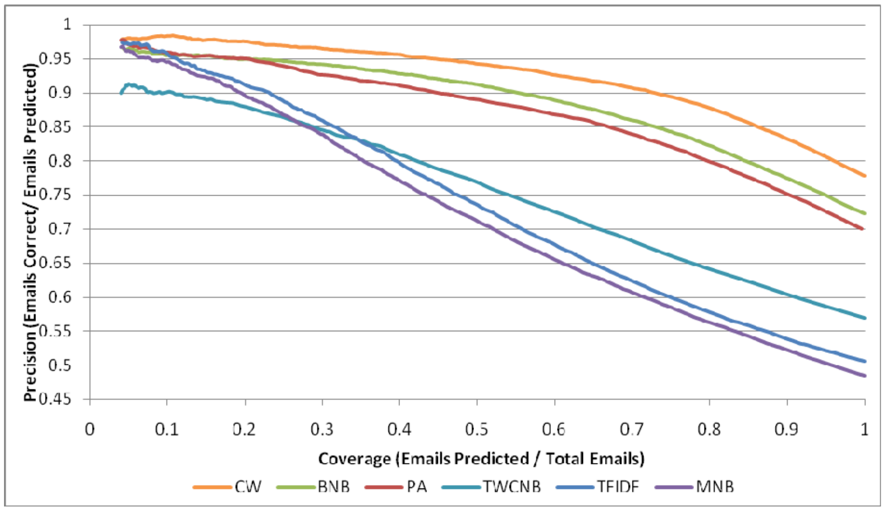

Figure 2 – Precision versus coverage graph for predictions made on the Full dataset, showing the decrease in precision as the confidence threshold is lowered.

As CW and BNB performed well in experiments both with and without email bodies, we composed the results from both data sets into the graph in Figure 3 to show how inclusion of email body affects the accuracy of the predictors. Interestingly, instead of improving across the board with the inclusion of additional data from the email bodies, the results are mixed. On NoBody accuracy of BNB increases throughout the range of coverage compared to Full. While CW has higher accuracy at 100% coverage on Full, at lower coverage CW performs better without the bodies. The other algorithms have similarly mixed results, some improving with the use of the bodies and others not.

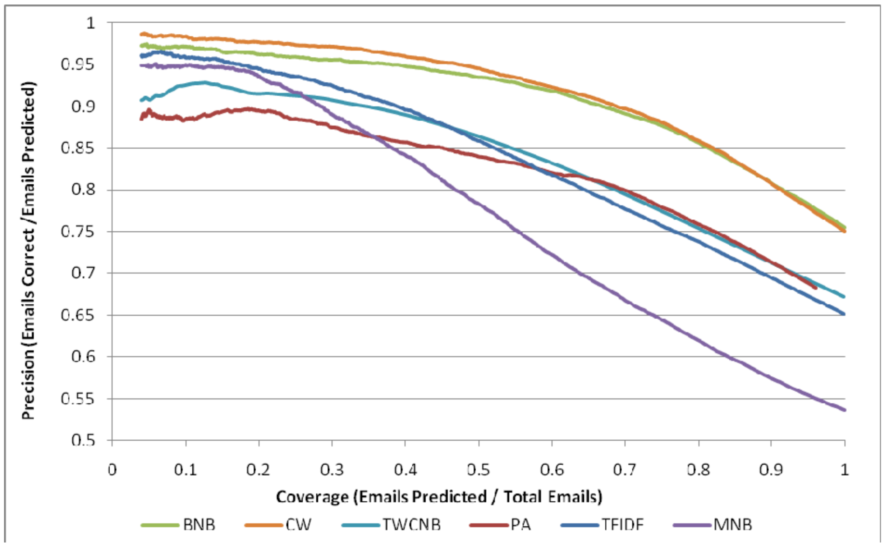

Figure 3 – Precision versus coverage graph for predictions made on NoBody.

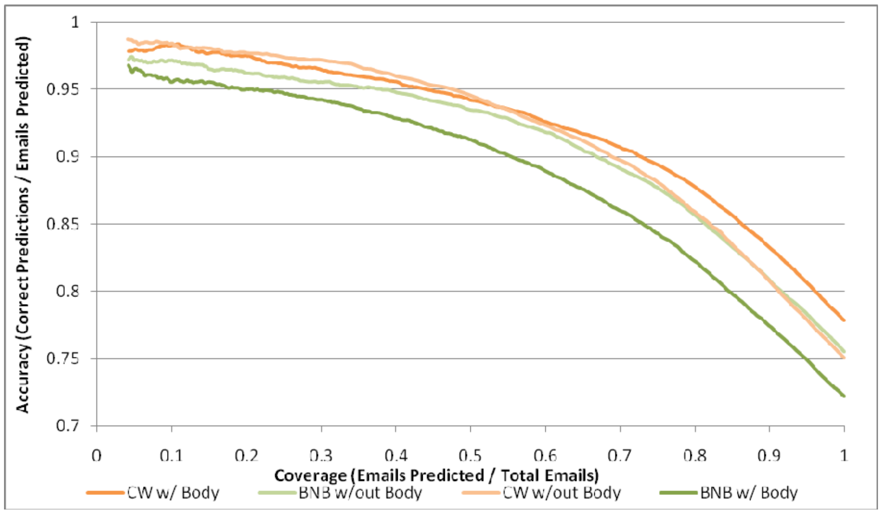

Figure 4 – Composed precision versus coverage graph of predictions on email data with email body included or not included on the top performing algorithms, to show the difference attributable to including email body.

These results indicate that CW with the bodies gives the highest precision. In particular, if we are interested in 90% precision, CW+Full can achieve this with 72% coverage, whereas BNB+NoBody can only achieve this with 67% coverage.

The number of classes in our data set, 380, is larger than all previous email classification studies. To determine to what extent the number of classes affects the performance of each of the different algorithms, we created several smaller subsets of the data in which we retain only data from very populous or very sparse classes. The

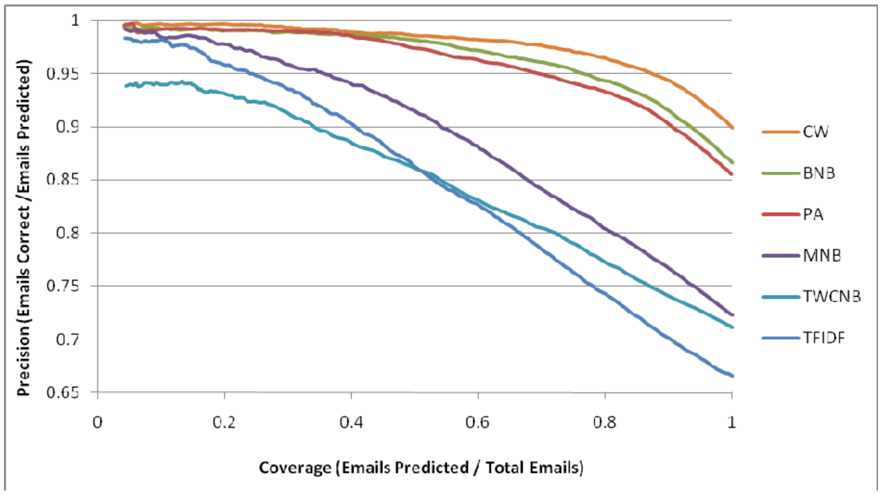

Figure 5 – Precision versus coverage graph for the 25 most populous classes in Full.

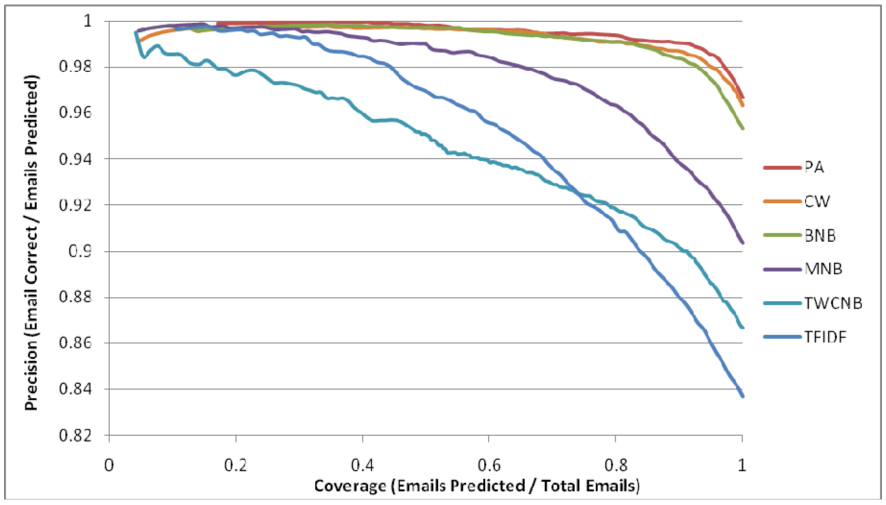

Figure 6 – Precision versus coverage graph for the 5 most populous classes in Full.

results on the 25 and 5 most populous classes for the Full dataset appear in Figures 5 and 6. Figures 7, 8, and 9 display the results of the 100, 25 and 5 most populous classes on the NoBody data set. We also look at the 100 least populous classes that have at least two examples. The results on Full and NoBody are displayed in Figures 10 and 11.

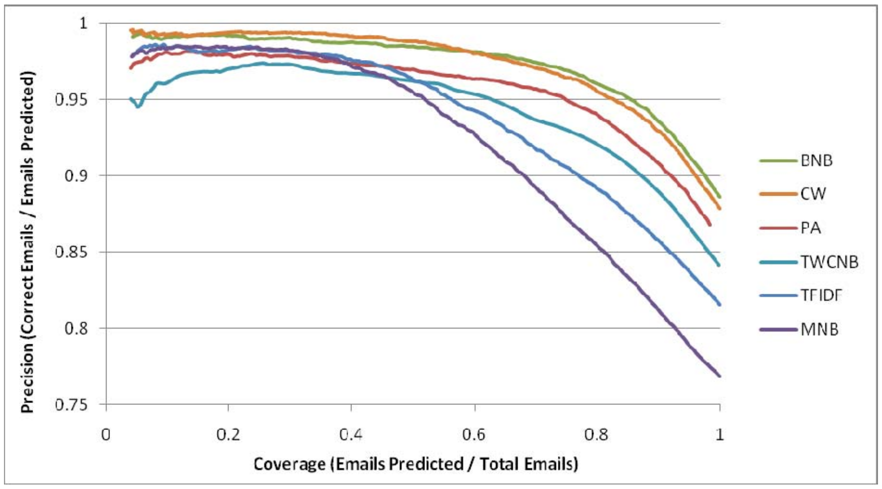

Figure 7 – Precision versus coverage graph for the 100 most populous classes in NoBody.

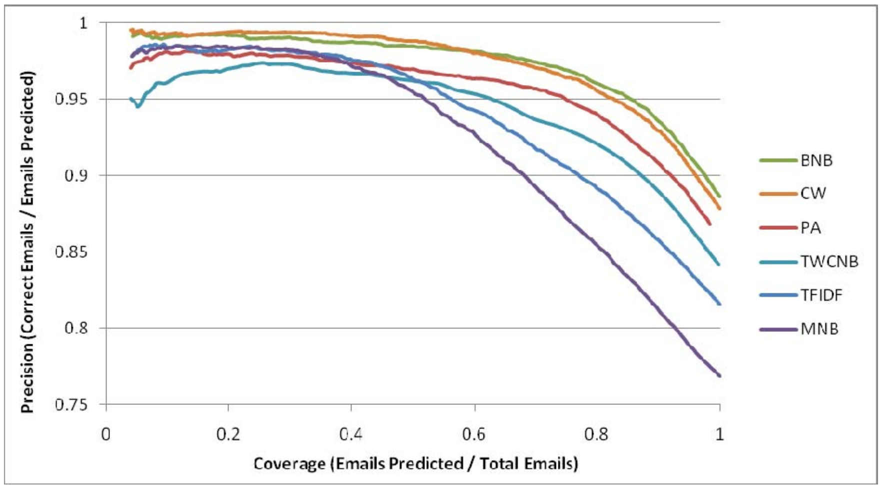

Figure 8 – Precision versus coverage graph for the 25 most populous classes in NoBody.

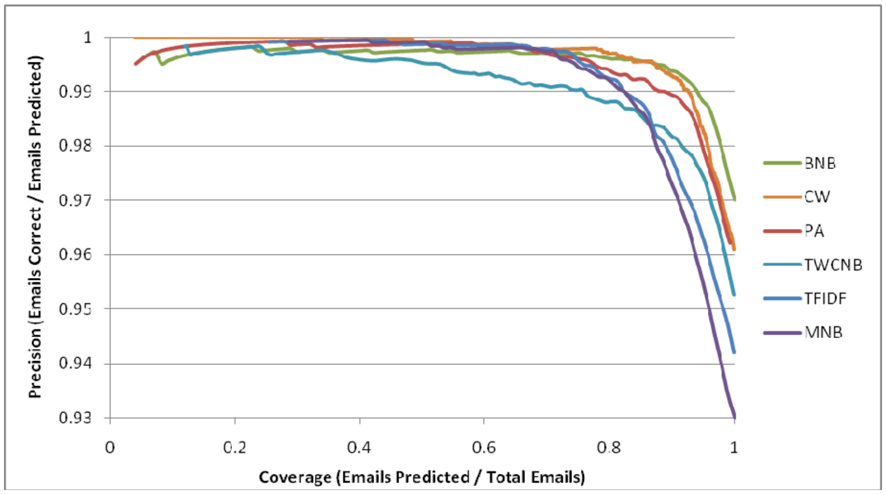

Figure 9 – Precision versus coverage graph for the 5 most populous classes in NoBody.

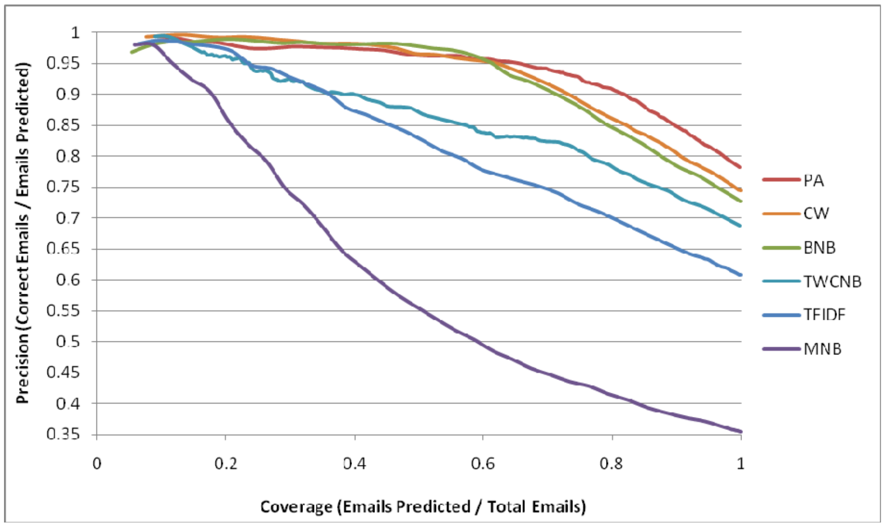

Figure 10 – Precision versus coverage graph for the 100 least populous classes (with >1 example) in Full.

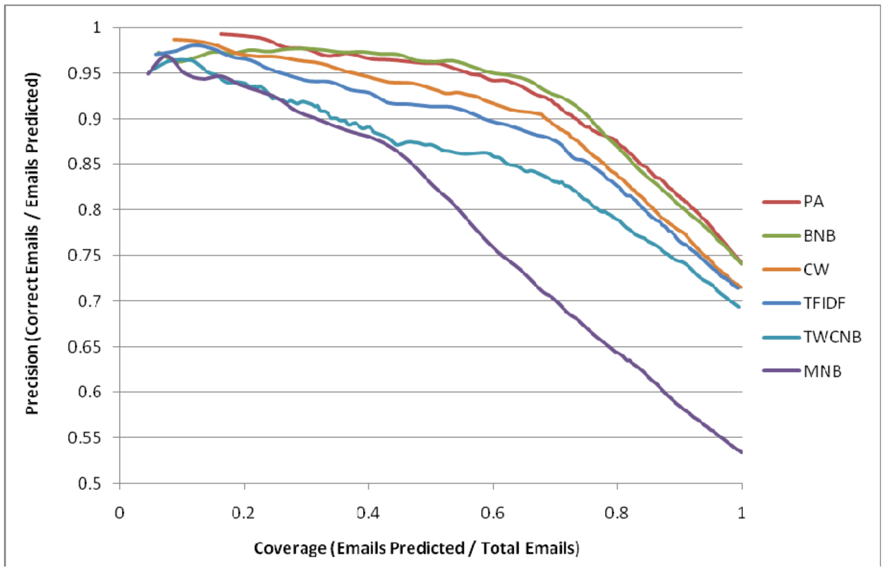

Figure 11 – Precision versus coverage graph 100 least populous classes (with >1 example) in NoBody.

As expected, all algorithms improve when measured on only the most populous classes. Their relative ordering is essentially the same as on the complete datasets, but some algorithms perform relatively better. For example, PA outperforms all of the other algorithms on the 5 largest classes in Full. This suggests that PA is more sensitive to large numbers of classes than the other algorithms. This makes sense, because the version of multiclass PA that we are using just learns a global reweighting of the class-specific TFIDF scores. This apparently breaks down when there are large numbers of classes. In addition, the TFIDF statistics are less reliable for very sparse classes. BNB's lead on CW also increases slightly with the decreasing number of classes on NoBody, and it also grows closer to the level of CW even on Full, possibly indicating that CW also has a slight weakness on large numbers of classes.

In the datasets with the 100 least populous classes, the results were very similar to the overall results and to the results of the 100 most populous classes in ordering, but with some differences and a reduction in accuracy. Most significantly, PA is the top-performing algorithm, improving in accuracy from the overall results. This is consistent with PA's early strong performance in the progressive results, since fewer training examples are seen in both cases, lending strength to the conclusion that PA is able to predict accurately on classes with very little training.

Figures 12 and 13 show the cumulative accuracy of each classifier for Full and NoBody. On Full, CW begins with a lead, which it maintains throughout the entire course of examples, showing that CW learns relatively quickly in addition to its ability to use the bodies of the email. Backing this conclusion up, on NoBody, CW still shows a very slight initial lead in accuracy, despite BNB's overall lead on NoBody. PA also demonstrates an ability to learn quickly on Full—it is more accurate than BNB for the first 5000 examples before BNB eventually overtakes it. Even on NoBody, PA shows an early advantage over TFIDF and TWCNB after a few hundred examples, which lasts until several thousand examples have been seen. This strengthens the argument that PA learns quickly, despite disadvantages in the long run.

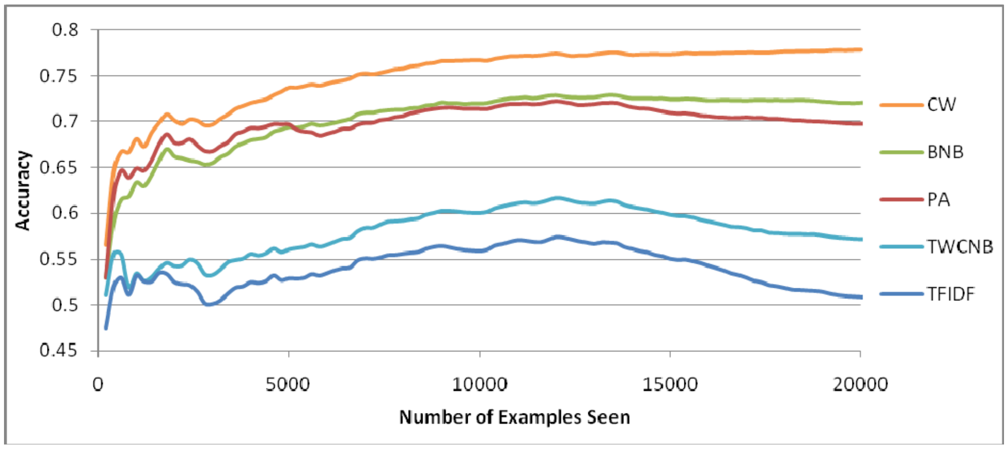

Figure 12 – Progressive results graph comparing how many examples have been seen to the cumulative accuracy to that point on the Full dataset.

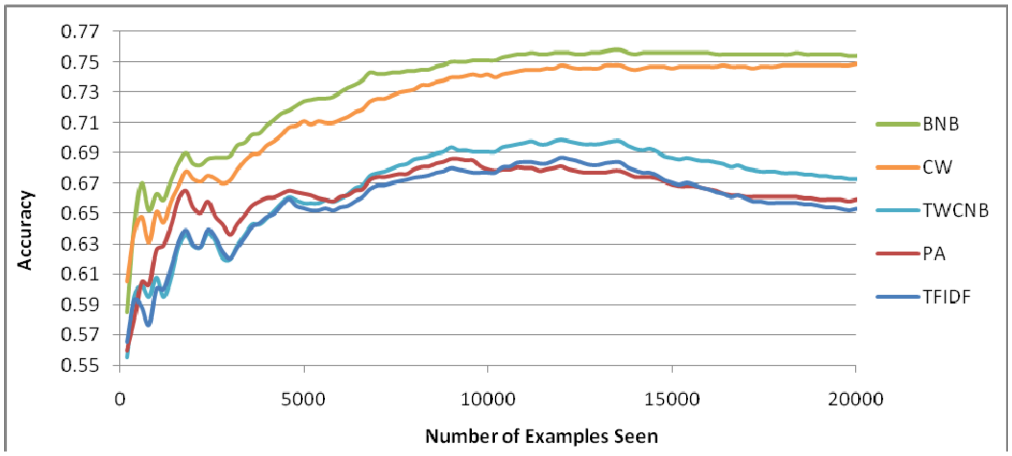

Figure 13 – Progressive results graph comparing how many examples have been seen to the cumulative accuracy up to that point on the NoBody dataset.

Several of the algorithms also demonstrate a gradual decline in accuracy, which seems counterintuitive since they are continually receiving additional training data. We determined that this is likely due to the addition of new classes after significant training had already occurred. Figures 14 and 15 show how the progressive results change when classes which first appear after 10,000 examples have already been seen are excluded on the Full and NoBody dataset. These graphs demonstrate a significantly reduced downward tendency with the exclusion of the later classes.

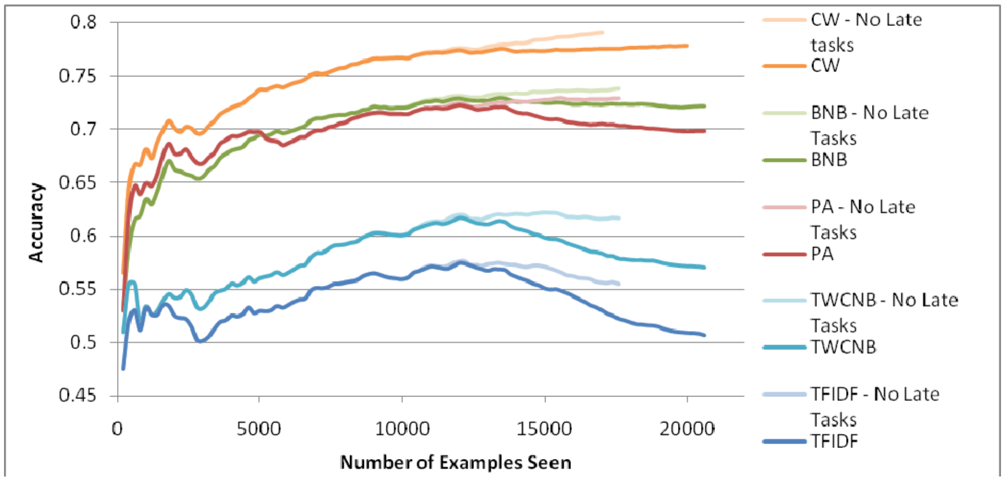

Figure 14 – Progressive results graph comparing how many examples have been seen to the cumulative accuracy up to that point on the Full dataset both with and without tasks which are introduced after 10,000 examples.

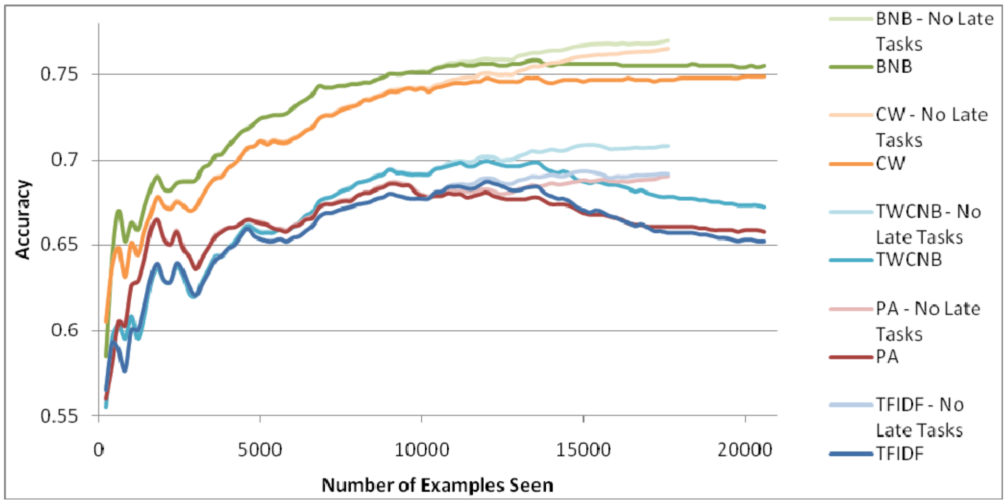

Figure 15 – Progressive results graph comparing how many examples have been seen to the cumulative accuracy up to that point on the NoBody dataset both with and without tasks which are introduced after 10,000 examples.

To show that this tendency is specific to classes which are introduced late and not simply to large numbers of classes, Figures 16 and 17 show the progressive results for the 25 most populous classes and for these same classes with those after the first 10,000 examples in the full dataset omitted. The results with the full 25 classes show a significant downward tendency, while the results with the later classes excluded show a reduced tendency, reinforcing that the problem lies in classes being introduced late, regardless of their size. CW and BNB, both of which show less of this tendency towards declining accuracy, seem to be more resilient against late introduced tasks.

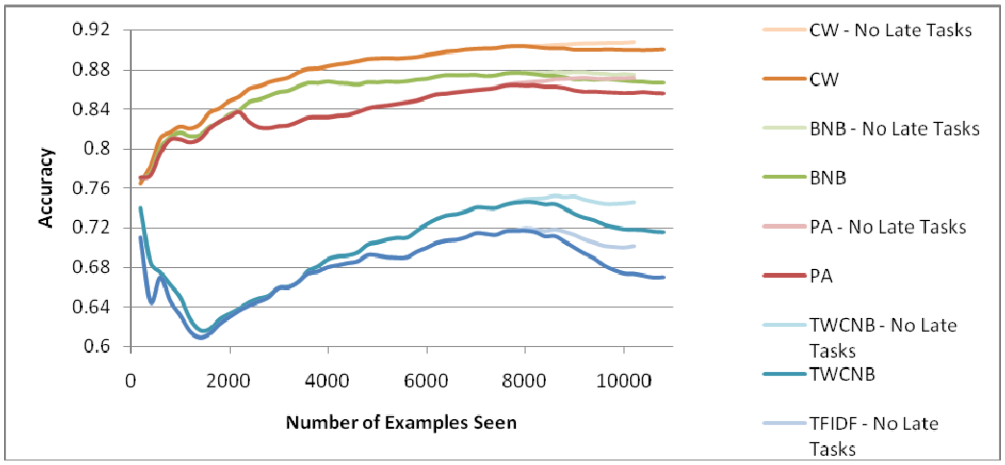

Figure 16 – Progressive results graph comparing number of examples seen to the cumulative accuracy up to that point on the 25 most populous classes of Full dataset both with and without tasks which are introduced after 10,000 examples.

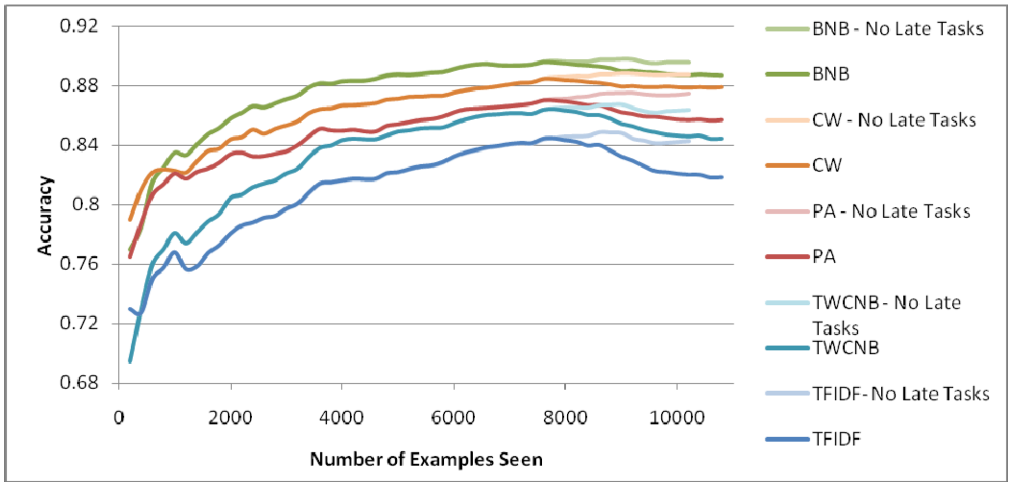

Figure 17 – Progressive results graph comparing number of examples seen to the cumulative accuracy up to that point on the 25 most populous classes of NoBody both with and without tasks which are introduced after 10,000 examples.

With classes ranging in number of examples from 1 to more than 2500, there is a large variation in how much training each class receives. Previous results from the progressive and number of classes investigations suggest that the size of classes is significant. We examine how the size of each class affects algorithm accuracy by tracking online the average (instantaneous) size of the true class for correct predictions and incorrect predictions, as well as the average (instantaneous) size of the incorrectly predicted class for incorrect predictions. The results are displayed in Table 1. The results show that MNB and TFIDF have a much higher average size for predictions made—they like to predict the popular classes. TWCNB also shows this same tendency, but to a lesser extent, which makes sense given that this is one of the problems that Rennie's modifications are supposed to overcome. BNB and CW have relatively even average sizes between the correct task and predicted task in incorrect predictions, showing resilience to class size in predictions made.

Table 1 – Average sizes of classes at the time that predictions are made for each of the different algorithms.

	Avg Size of Correctly Predicted Class	Avg Size of Correct Class in Incorrect Predictions	Avg Size of Predicted Class in Incorrect Predictions
TFIDF	346	81	532
PA	310	145	314
BNB	280	171	196
MNB	413	68	682
TWCNB	333	91	354
CW	288	146	195

Our final analysis focuses on the problem of repeated errors. As mentioned above, in the current TaskTracer system, we sometimes observe that the email predictor seems to have a "favorite class of the day" such that it repeatedly predicts a particular class *c* regardless of the true class of the email message. We also notice cases where all messages belonging to class *c* are repeatedly misclassified as various other classes.

This pattern of repeated error persists even as the user is continually giving corrective feedback. Users hate this kind of behavior.

To determine if algorithms were making predominately the same mistakes, we define three kinds of repeated errors. *Repeated source* errors occur when two messages both belong to true class *c* but are both misclassified (to the same or different predicted classes). *Repeated destination* errors occur when two messages are both incorrectly predicted to belong to class *c*. Finally, *Repeated Confusion* errors occur when two messages that both belong to true class *c* are both predicted to belong to a different class *c'*. Figures 18, 19, and 20 plot the number of repeated errors as a function of the temporal separation between the pairs of email messages. Specifically, if two misclassified messages are within window size *W* messages of each other, then they are included in the plot for the corresponding type of repeated error.

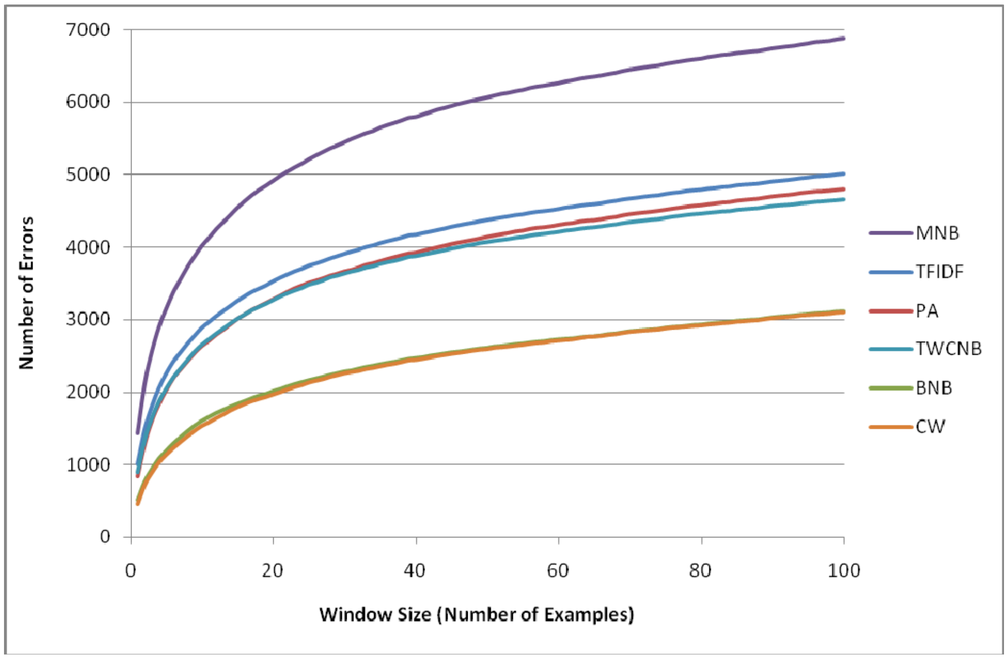

Figure 18 – Repeated error graph showing the number of repeated source errors, in which two emails from the same class are both predicted incorrectly within a window of examples, compared to the size of the window for each of the different algorithms.

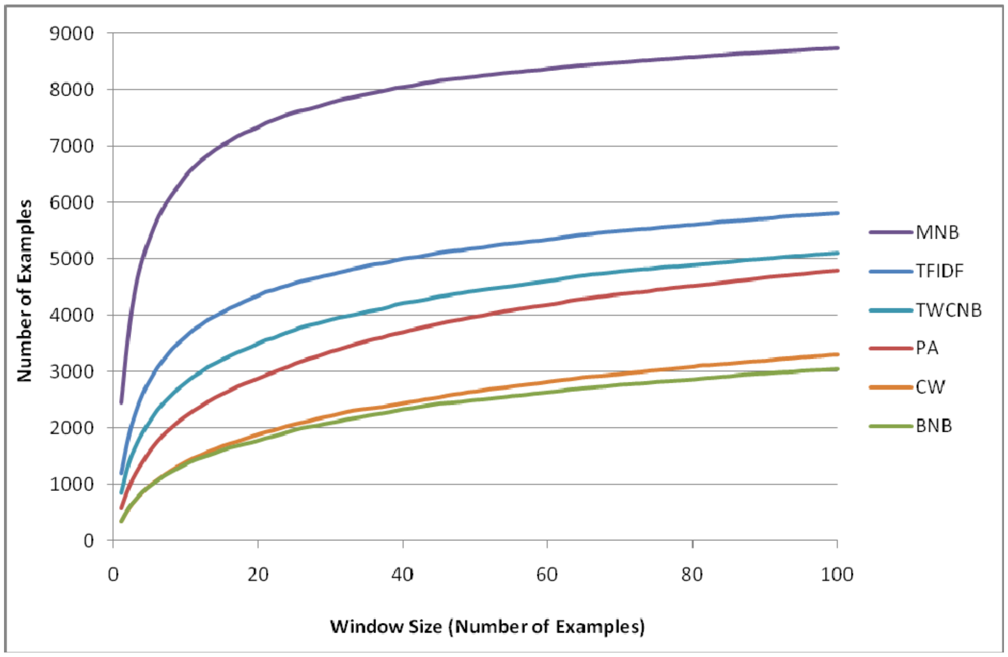

Figure 19 – Repeated error graph showing the number of repeated destination errors, in which two emails are both incorrectly predicted to be of the same class within a window of examples, compared to the size of the window for each of the different algorithms.

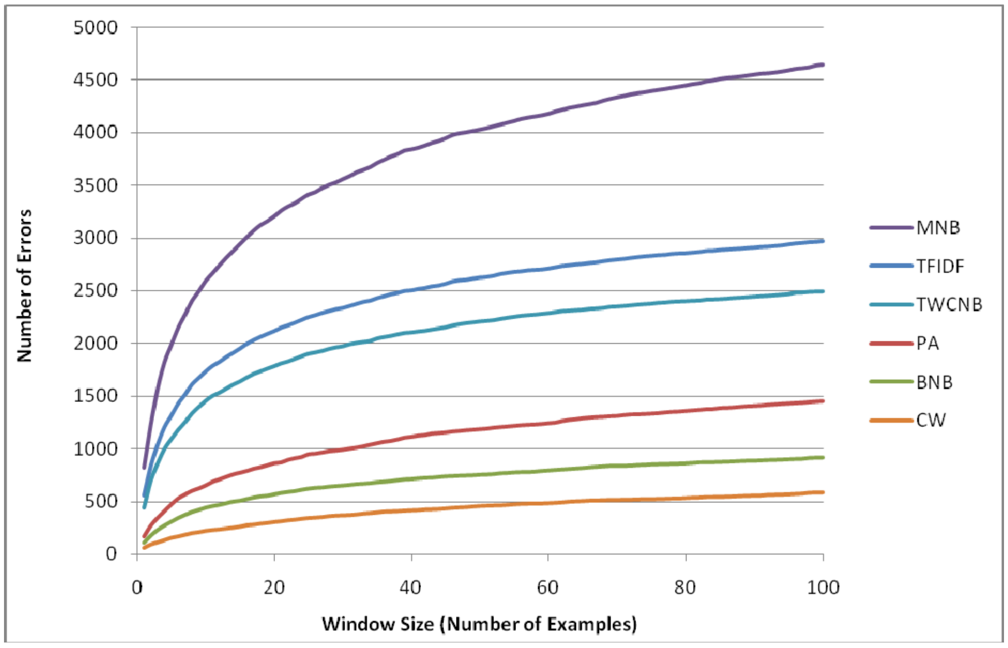

Figure 20 – Repeated error graph showing the number of repeated errors sharing both source and destination, in which two emails labeled with the same class are both incorrectly predicted to be of the same class within a window of examples, compared to the size of the window for each of the different algorithms.

These graphs show that repeated destination errors are more common than repeated source errors, likely due to large classes being assigned to emails that belong to many different smaller classes. The graphs also show that MNB, TFIDF and TWCNB are significantly more likely than the other algorithms to make repeated errors. For repeated source and repeated destination errors, BNB and CW are essentially tied for first place. CW is the clear winner for repeated confusion errors. The results suggest that multiclass PA—despite its attractive theoretical basis—does not do all that well at avoiding repeated errors. But CW’s performance on this measure is impressive.

7. CONCLUSION

The Confidence Weighted and Bernoulli Naïve Bayes algorithms show the most promise for email classification, with CW generally giving the best performance. Both perform very well on our email data set, showing good performance on both frequent and sparse classes. They also both avoid repeated errors. Conversely, Multinomial Naïve Bayes and Term Frequency-Inverse Document Frequency show themselves to be poor choices. The generality of these conclusions is limited by the fact that we only have data from one user, but the size and complexity of this data set provide a good basis for eliminating some algorithms from further consideration. We plan to deploy CW, BNB, and PA in a publically-distributed version of TaskTracer later this year.

REFERENCES

[1] Bekkerman, R., McCallum, A., and Huang, G. Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora. *CIIR Technical Report IR-418,* UMass Amherst, 2004.

[2] Cohen, W. W. Learning rules that classify e-mail. In *Proceedings of AAAI Spring Symposium on Machine Learning and Information Retrieval,* 1996.

[3] Crammer, K., Dekel, O.,Keshet, J., Shalev-Shwartz, S. and Singer, Y. Online Passive-Aggressive Algorithms. *JMLR 7* (2006), 551-585.

[4] Dietterich, T., Slater M., and Bao, X.. Quantifying and Supporting Multitasking for Intel Knowledge Workers. 2009.

[5] Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., Herlocker, J. L. TaskTracer: A Desktop Environment to Support Multi-tasking Knowledge Workers. *IUI2005*, 75-82.

[6] Dredze, M., Crammer, K., and Pereira, F. In *ICML2008*, (Helsinki, Finland) 2008.

[7] Kiritchenko, S. and Matwin, S. Email classification with co-training. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research*, 2001.

[8] Provost, J. Naive-Bayes vs. rule-learning in classification of email. Technical Report *AI-TR-99-284*, University of Texas at Austin, Artificial Intelligence Lab, 1999.

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

[10] Rennie, J.D.M., Shih, L., Teevan, J. and Karger D.R. Tackling the Poor Assumptions of Naïve Bayes Text Classifiers. In *ICML2003* (Washington D.C.), 2003.

[11] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.

[12] Segal, R. and Kephart, J. Incremental learning in swiftfile. In *ICML2000,* 2000.

[13] Shen, J., Li, L., Dietterich, T., and Herlocker, J. A Hybrid Learning System for Recognizing User Tasks from Desktop Activities and Email Messages. *IUI2006,* 86-92 (Sydney, Australia), 2006.

.