Naïve-Bayes vs. Rule-Learning in Classification of Email

Jefferson Provost
Department of Computer Sciences
The University of Texas at Austin
jp@cs.utexas.edu

Abstract

Recent growth in the use of email for communication and the corresponding growth in the volume of email received has made automatic processing of email desirable. Two learning methods, naïve bayesian learning with bag-valued features and the RIPPER rule-learning algorithm have shown promise in other text categorization tasks. I present three experiments in automatic mail foldering and spam filtering, showing that naïve bayes outperforms RIPPER in classification accuracy.

1 Introduction

The volume of email that we get is constantly growing. Most modern mail reading software packages provide some form of programmable automatic filtering, typically in the form of sets of rules that file or otherwise dispose of mail based on keywords detected in the headers or message body. Unfortunately programming these filters is an often arcane and sometimes time-consuming process. An adaptive mail system which can learn its users' mail sorting preferences would be very attractive.

This paper compares two learning algorithms, one probabilistic and one rule-based, on the task of learning to sort or filter email. Both algorithms have been used previously with success in text categorization tasks, but this paper will show that the probabilistic algorithm shows more promise than the rule-learner for email classification.

2 Email Classification

2.1 Data Representation

In the experiments in this paper email messages are represented in a structured "bag of words" representation. Each email message is represented as vectors of features, in which the message body and each individual message header are represented as separate features. The content of each feature is all the words that appear in that feature, with repeated words counted multiple times.

In tokenizing the messages, all letters are converted to a single case, all punctuation is removed, and email addresses, domain names, URLs, etc, are broken down into their constituent "words." For example

```
From:  Jefferson Provost
   <jp@cs.utexas.edu>
```

would become

$$from = \{jefferson, provost, jp, cs, utexas, edu\}.$$

Many high-frequency, low-information content words, such as "a," "an," "the," most prepositions and conjunctions, and all single-character words are removed from the token stream before bagging, however, no complex stemming is performed.

2.2 Algorithms

The experiments in this paper compare two learning algorithms: a naïve bayesian algorithm used by Mooney et al. (1998) for text categorization, and RIPPER, a rule-learning approach used by Cohen (1996) for categorization of email

Rule Learning The rule learning algorithm used in these experiments is the RIPPER algorithm described by Cohen (1995). It is a propositional learner designed for efficient performance on large, noisy datasets. RIPPER is designed to handle set- and bag-valued attributes equivalently by generating what Cohen calls "keyword-spotting rules." These are rules of the form

$$cs328 \leftarrow \text{“}utexas'' \in from \wedge \text{“}utexas'' \in to.$$

This rule states that a message belongs in the folder *cs328* if the word "utexas" appears in both the *from* and *to* headers. Rules like this are highly suitable for email classification and filtering because many email reading programs are already equipped to use rules of this type in classification, thus integrating this kind of rule-learning system into existing mail systems becomes a question of converting the syntax of the rules into one understood by the mail reader.

Naïve Bayes The second learning algorithm is a feature-based bayesian text classfier similar to the one described in Mooney et al. (1998), but extended to handle bag-valued features. The ability of this classifier to utilize the word counts in the bags of words in calculating its probability tables should give it an advantage in classification accuracy over ripper. As the experiments will show, this seems to be the case.

The disadvantage of a bayesian classifier is difficulty of integration with existing mail reading software, because of the lack of a rule-based representation of the classification. This may be changing, however. For example, the *Gnus* news and mail reading system (Ingebrigsten 1999), distributed with

University of Texas at Austin, Artificial Intelligence Lab. Technical Report AI-TR-99-284.

GRACS:	108	5.27%
ROBOT:	175	8.53%
PSYSCOPE:	109	5.31%
NN:	131	6.39%
SCHOOL:	69	3.36%
CS328:	691	33.69%
PERSONAL:	694	33.84%
CONNECTIONISTS:	74	3.61%
Total examples:	2051	100.00%

Table 1: **Data-set 1: Hand Sorted Mail.** The categories and distribution for data set 1. These data comprise 4 months of email, hand-sorted by the author.

recent versions of *GNU Emacs* has hooks that allow installation of arbitrary programs for filtering and foldering news and mail. Furthermore, there are several open-source mail-readers which could be modified to include a hooks for arbitrary classifiers.

3 Experiments

Below I will describe experiments comparing these two systems in three tasks: learning a user's foldering preferences from his hand-sorted email, reconstructing the categorization policy of a hand-coded automated mail classifier from a set of machine-sorted email, and learning to identify junk email from a set of messages classified as either spam, or not. All experiment were run using ten-fold cross-validation. Learning curves and significance measures from two-tailed t-tests are provided. As a baseline comparison, the accuracy that would be achieved by always choosing the most common category in the training data is also plotted for each experiment.

3.1 Hand-Sorted Mail

The first experiment compares the accuracy of the classifiers at learning a user's mail sorting preferences from sorted mail. The input data are four months of the author's sorted mail. Table 1 shows the folders and distribution of messages in the data set.

These data pose an interesting challenge for a learning system. Not only is the distribution of messages in to folders highly nonuniform, but the selection of folders for messages is strongly ideosynchratic. While the contents of one folder, CONNECTIONISTS, are entirely determined by a single keyword match ("$connectionists'' \in to$), the rest were not deterimined by a single keyword match on the *from* or *to* fields, but rather by my judgement of what folder would the best mnemonic for later retrieval of the message based on its content, sender and recipients. In this case, the task of the learner is to induce a model of the user's mail sorting preferences.

The results of this experiment are shown in Figure 1. Naïve bayes performs significantly better ($p < 0.01$) than RIPPER throughout learning, achieving 87% test accuracy after 400 training examples, and 80% test accuracy after 175 examples.

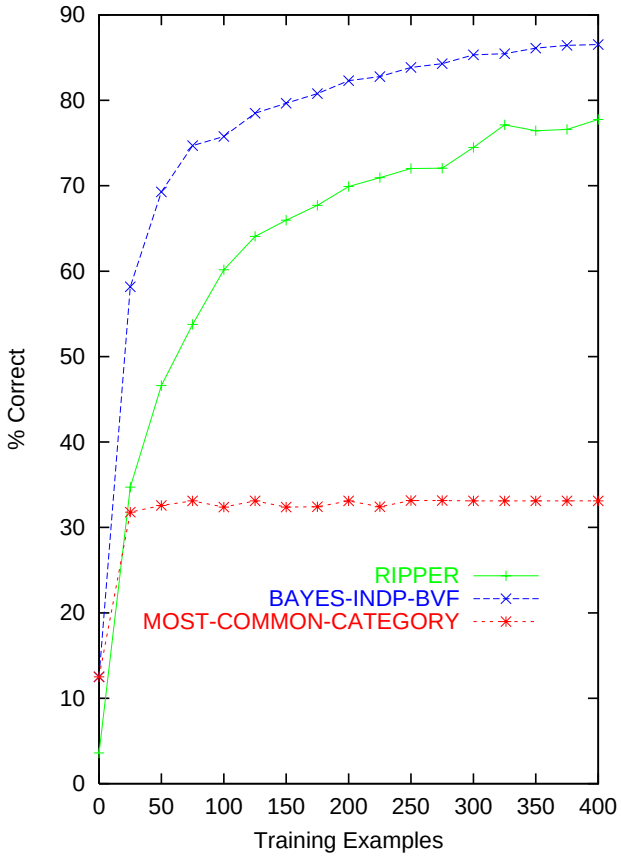

Figure 1: **Test Accuracy on Hand-Sorted Data.** Naïve Bayes significantly outperforms ($p < .01$) RIPPER on dataset 1: hand sorted email.

RIPPER achieves 78% accuracy after 400 examples, and only 67% accuracy after 175 examples.

3.2 Automatically-Sorted Mail

The second experiment compares the two classifiers on a data set in which each folder consisted entirely of messages from a single mailing list of users or developers of a Debian/GNU software package. These messages were automatically sorted by a hand-crafted classifier consisting of pattern-match rules. Some of the handcrafted rules contain regular expressions, and thus are somewhat more complex than simple keyword-spotting rules. Nevertheless, it is reasonable to expect that both naïve bayes and RIPPER will do well on this dataset. Table 2 shows the categories and distribution of the data set. Test accuracy of each learner on these data is shown in Figure 2.

As expected, both classifiers perform well on this data set. Naïve bayes, however, outperforms RIPPER by a slight but statistically significant margin ($p < .05$) when trained on 175 examples or more.

GNOMEANNOUNCE:	302	1.74%
LJFS:	101	0.58%
VORTEX:	398	2.29%
ALSAUSER:	2728	15.71%
TULIP:	1137	6.55%
BBDB:	332	1.91%
DINGGNUS:	3681	21.20%
DEBIANMENTORS:	3377	19.45%
AMANDAUSERS:	5304	30.55%
Total examples:	17360	100.00%

Table 2: **Data-set 2: Autmatically Filtered Mail.** The categories and distribution for dataset 2. These data consist of messages from Debian/GNU user and developer mailing lists that were automatically sorted into folders by hand-built rule-based classifier.

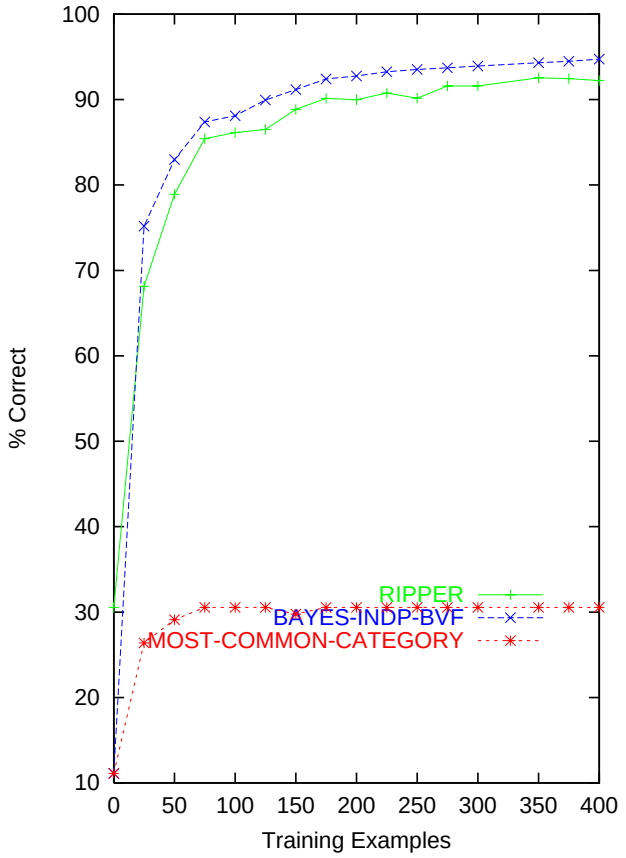

Figure 2: **Test accuracy on Machine-Sorted Email.** Naïve Bayes shows a small advantage in accuracy even on machine sorted messages. The results are significant at $p < 0.05$, for training on 175 examples or more.

Spam:	204	22.7%
Non-spam:	694	77.3%
Total examples:	898	100.0%

Table 3: **Data-set 3: Spam Data.** This dataset consists of spam messages donated from a several sources, and non-spam messages comprised of a subset of dataset1.

3.3 Spam Detection

The final experiment test the ability of the two algorithms to learn to detect junk email, or spam, from the content of the messages. In this dataset, the junk examples are a collection of junk mail donated by several users, while the non-junk messages are a subset of the data used in the first experiment. For this reason, the *From:* and *To:* headers have been excluded from consideration in learning. I do not believe this to be a problem, however, because junk mail advertisers typically forge both the *From:* and *To:* headers in an attempt to both disguise the origination point of the spam and fool automatic spam filters. Thus it is reasonable to that the learners should focus on the content of the message, namely the *Subject:* header and the message body. The distribution of the data is shown in Table 3.

Figure 3 shows that again the naïve Bayesian classifier significantly ($p < .001$) outperforms RIPPER in classification accuracy. Naïve bayes is recognizing spam with 90% accuracy after training on 25 examples and has reached 95% accuracy after 50 examples. RIPPER meanwhile, is still struggling to reach 90% accuracy after 400 training examples.

4 Discussion and Future Work

4.1 Methodology

The above experiments show that a naïve bayesian classifier with bag-valued features is a promising method for automatically constructing email classifiers, with the obvious caveat that the experiments were run on only a limited sample of data. Unfortunately more studies with samples of several users' foldering preferences are difficult to perform because of users' understandable reluctance to donate private email for study. Furthermore it is questionable to what extent "abridged" collections of email will be useful in truly assessing the system's ability to model users' foldering preferences.

4.2 Classification Accuracy in Foldering

It remains to be seen whether The 87% accuracy achieved on hand-sorted email in the first experiment is typical of what can be achieved with the average user. If it is, however, it does not seem accurate enough to be confidently installed for automatic pre-sorting of email. There are a some ways in which this might be overcome, however. First, a mail classifier could be made to only pre-sort email for which it has high confidence in is choice of folders, while the messages for which it is unsure are placed in the user's INBOX or default folder, to be handled by the user. Then the user's foldering

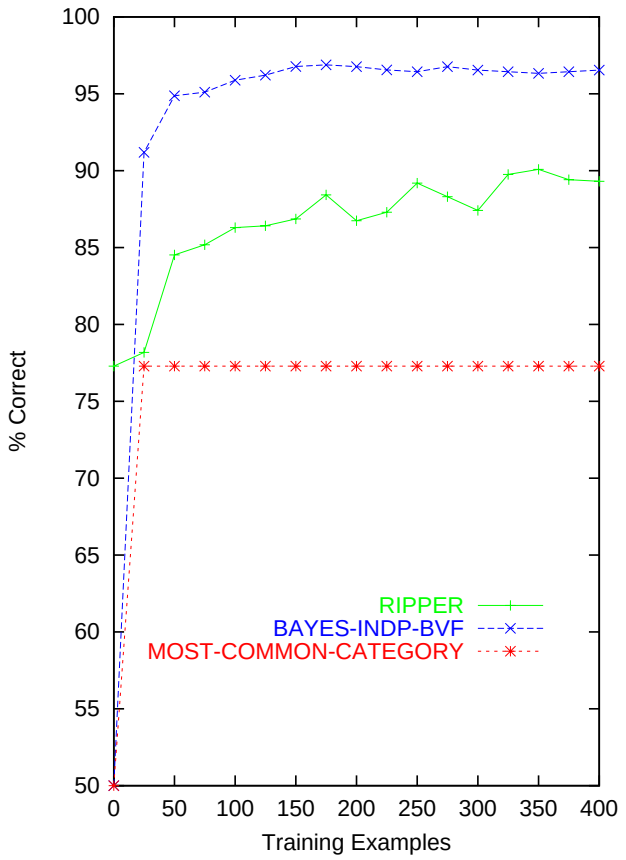

Figure 3: **Test Accuracy in Spam Detection.** Naïve Bayes significantly outperforms RIPPER ($p < .001$) in detecting spam from message content (*Subject:* and *message body* only).

choices on these messages could be fed back to the learner to refine its hypothesis. Further, it may not always be necessary to pre-folder at all; many mail readers, such as *Gnus*, and *Pine* offer a default choice (e.g. the last folder used) when the user chooses to file a message in a folder. A system that automatically chose the correct result 87% of the time would be very useful.

4.3 Future Work in Spam Detection

Spam detection and filtering both for email and USENET messages seems to be the most promising area of application suggested by these experiments. Direct marketers and system adminstrators are in a race of constant adaptation as the system administrators implement new schemes for detecting and blocking spam while the spammers find new ways to evade being blocked so that they can reach their potential customers. A fast learning, accurate, adaptive spam catcher would be a valuable tool for system administrators and users wishing to block spam in email and USENET messages.[1]

[1]It could also be a tool for the spammers, who could use it to test new potential spam for "undetectability." Such is the way of an arms race.

Before such a tool can be put in place, however, more investigation must be done. In particular, these experiments did not investigate the precision and recall of either classification method. A good blocking device, however, should be sensitive to the cost of different kinds of misclassification errors, and tunable with respect to them. While many users would rather let a few spam through than miss legitimate messages, some users may prefer just the opposite.

5 Conclusion

I have presented three experiments comparing a naïve Bayesian algorithm with bag-valued features against the RIPPER rule learning algorithm in different email classification tasks. In learning a user's foldering preferences, and learning to detect spam, the Bayesian classifier substantially outperformed RIPPER in classification accuracy. In reconstructing the policy of an automated, rule-based email classifier, both systems performed very well, but the Bayesian classifier still showed a small but statistically significant improvement over RIPPER.

References

Cohen, W. W. (1995). Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference*.

Cohen, W. W. (1996). Learning rules that classify e-mail. In *AAAI Spring Symposium on Machine Learning in Information Access*.

Ingebrigsten, L. M. (1999). Gnus network user services. World Wide Web Site: http://www.gnus.org/.

Mooney, R. J., Bennett, P. N., and Roy, L. (1998). Book recommending using text categorization with extracted information. In *Papers of the AAAI-98/ICML-98 Workshop on Learning for Text Categorization and Papers of the AAAI-98 Workshop on Recommender Systems*. Madison, WI.

Mooney, R. J., and Roy, L. (1999). Content-based book recommending using learning for text categorization. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evalutation*. Berkely, CA.