Learning Rules that Classify E-Mail

William W. Cohen
AT&T Laboratories
600 Mountain Avenue Murray Hill, NJ 07974
(908)-582-2092
wcohen@research.att.com

Abstract

Two methods for learning text classifiers are compared on classification problems that might arise in filtering and filing personal e-mail messages: a "traditional IR" method based on TF-IDF weighting, and a new method for learning sets of "keyword-spotting rules" based on the RIPPER rule learning algorithm. It is demonstrated that both methods obtain significant generalizations from a small number of examples; that both methods are comparable in generalization performance on problems of this type; and that both methods are reasonably efficient, even with fairly large training sets. However, the greater comprehensibility of the rules may be advantageous in a system that allows users to extend or otherwise modify a learned classifier.

Introduction

Perhaps the most-discussed technical phenomenon of recent years has been the rapid growth of the Internet—or more generally, the rapid growth in the number of on-line documents. This has led to increased interest in intelligent methods for filtering and categorizing documents. For example, there is now a great deal of interest in systems that allow a technically naive user to easily construct a personalized system for filtering and classifying documents such as e-mail, netnews articles, or Web pages.

This paper will compare methods for learning text classifiers, focusing on the kinds of classification problems that might arise in filtering and filing personal e-mail messages. A particular focus will be on learning plausible message categories from relatively small sets of labeled messages.

The goal in performing this study is to determine how "traditional" IR learning methods compare with systems that learn concise and easily-interpreted classifiers; specifically, to compare traditional IR methods with systems that learn sets of *keyword-spotting rules*.

In a "keyword-spotting" rule the primitive conditions test to see if a word appears (or does not appear) in a certain field of a mail message. An example of a set of keyword-spotting rules is below.

cfp $\leftarrow$ "cfp" $\in$ subject, "95" $\in$ subject.
cfp $\leftarrow$ "cfp" $\in$ subject, "1995" $\in$ subject.
cfp $\leftarrow$ "call" $\in$ body, "papers" $\in$ body.

This rule set states that an e-mail message will be classified as an instance of the class "cfp" (call for papers) if it contains the tokens "cfp" and "95" in the subject field, or the tokens "cfp" and "1995" in the subject field, or if it contains the tokens "call" and "papers" in the body of the message.[1]

The motivation for learning keyword-spotting rules is based on first, a belief these rules will relatively easy for end users to understand and modify, and second, a suspicion that learning methods alone are not an adequate solution for categorization problems of this type. It seems likely that instead some mix of automatically and manually constructed classifiers will be necessary to account for the fact that both the user's interests and the distribution of messages change (sometimes quite rapidly) over time. For instance, at the time of this writing, the ruleset above may be accurate for messages I have received over the last few months; however, at some point it will certainly become appropriate to modify it by replacing "95" with "96" and "1995" with "1996".

Although keyword-spotting rulesets have the advantage of comprehensibility, to my knowledge they have not been extensively evaluated on text categorization problems. It should be noted that these rulesets are quite different from the classifiers constructed by more common text categorization learning methods, such as naive Bayes or term frequency/inverse document frequency weighting (TF-IDF) (Salton 1991). Rather than making decisions based on a weighted combination of all the words in a document, rules make decisions based on a small number of keywords. Also, keyword-spotting rules do not base classification decisions on word frequency, only on the presence or absence of a word.

One goal of our evaluation is to determine how much

[1]The learning algorithm considered here does *not* assume that keywords like "cfp" or "1995" are drawn from some small set. Instead, any token appearing in any training example is a possible keyword.

accuracy (if any) is lost by using keyword-spotting rules, relative to other classifiers. A second goal is to determine how much CPU time is needed to learn accurate rulesets on moderate-sized sets of examples—in particular, whether it is reasonable to use rule learning as a component of an interactive message-filtering system. A final goal is to gain some understanding of the number of examples necessary to learn accurate classifiers.

Learning algorithms

Two text categorization algorithms will be compared. The first uses TF-IDF weighting (Salton 1991). The implementation used here follows Ittner *et al.* (1995), who adapt Rocchio's relevance feedback algorithm (Rocchio 1971) to classification. Briefly, each document is represented as a vector, the components of which correspond to the words that appear in the training corpus. For a document d, the value of the component for the word w_i depends on the frequency of w_i in d, the inverse frequency of w_i in the corpus, and the length of d. Learning is done by adding up the vectors corresponding to the positive examples of a class C and subtracting the vectors corresponding to the negative examples of C, yielding a "prototypical vector" for class C. Document vectors can then be ranked according to their distance to the prototype. A novel document will be classified as positive if this distance is less than some threshold t_C, which can be chosen so as to balance recall and precision in some set manner. In our experiments, t_C was chosen to minimize error on the training set.

The second algorithm is an extension of the rule learning algorithm RIPPER, which is described in detail elsewhere (Cohen 1995a). Briefly, RIPPER builds a ruleset by repeatedly adding rules to an empty ruleset until all positive examples are covered. Rules are formed by greedily adding conditions to the antecedent of a rule (starting with an empty antecedent) until no negative examples are covered. After a ruleset is constructed, a optimization postpass massages the ruleset so as to reduce its size and improve its fit to the training data. A combination of cross-validation and minimum-description length techniques are used to prevent overfitting. In previous experiments, RIPPER was shown to be comparable to C4.5rules (Quinlan 1994) in terms of generalization accuracy, but much faster for large noisy datasets.

Before running these experiments, RIPPER was modified so as to be more efficient for text categorization problems. In the initial implementation of RIPPER, examples were represented as feature vectors. This implementation could be used to learn keyword-spotting rules; however, it would be necessary to construct a boolean feature for each possible condition of the form "$w_i \in$ *field*", and then to represent each document as a vector of these boolean features. This is rather inefficient since even a moderately large corpus will contain hundreds or thousand of words.

One common way of avoiding this problem is to restrict the vocabulary, for example by only considering frequent words or highly informative words. I chose instead to extend RIPPER to allow the value of an attribute to be a set of symbols (as well as single symbolic value or a number). This means that a structured document can be easily and naturally represented. For example, an e-mail message is represented with four attributes, `from`, `to`, `subject`, and `body`. The value of each attribute is the set of all words that appear in the corresponding section of the mail message.

The primitive tests on a set-valued attribute a (*i.e.*, the tests which are allowed in rules) are of the form "$w_i \in a$" or "$w_i \notin$ *field*". When constructing a rule, RIPPER finds the test that maximizes information gain for a set of examples S efficiently, making only one pass over S for each attribute. All words w_i that appear as elements of attribute a for some training example are considered by RIPPER.

Using set-valued attributes allows one to represent a set of documents easily and naturally. It is arguably more elegant and potentially more robust than using, say, entropy-based feature selection (Lewis and Ringuette 1994; Apté *et al.* 1994) to derive a small of features. This representation also simplifies the preprocessing of examples—a worthwhile aim if one's eventual goal is integration of the learner into an interactive system. Set-valued attributes are discussed in more detail elsewhere (Cohen 1996a).

A second extension to RIPPER, also motivated by text categorization problems, allows the user to specify a *loss ratio* (Lewis and Catlett 1994). A loss ratio indicates the ratio of the cost of a false negative to the cost of a false positive; the goal of learning is to minimize misclassification cost on unseen data. Loss ratios in RIPPER are implemented by changing the weights given to false positive errors and false negative errors in the pruning and optimization stages of the learning algorithm.

Recall that the TF-IDF learning algorithm can trade off recall for precision by making an appropriate choice of its similarity threshold t_C. By using an appropriate loss ratio RIPPER can also make this trade-off. The experimental comparisons discussed below, however, focus on classifier error; unless otherwise stated a loss ratio of 1 was used.

In all of the experiments described below, messages were parsed into a header and body. All words from the `from`, `to`, and `subject` fields were extracted from the header, and the first 100 words were extracted from the body. (A word is any maximal-length sequence of alphanumeric characters, normalized by converting uppercase to lower-case characters.) For RIPPER, examples were represented by four set-valued attributes. For TF-IDF, examples are represented by a set of tokens of the form f_w, where w is a word and f is the field w appeared in; for instance the word `call` appearing in

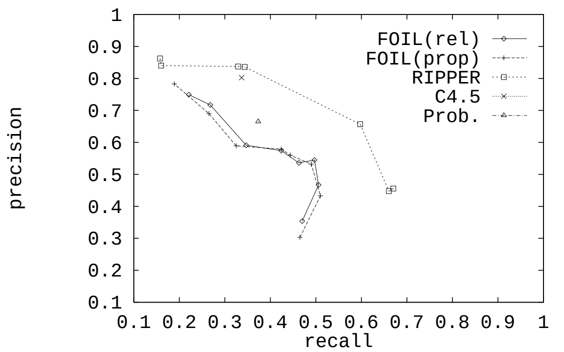

Figure 1: AP problems with uncertainty samples

the `subject` field would be encoded as `subject_call`. Header fields other than "to", "from", and "subject" were discarded. Any random sample that contained no positive examples of the concept to be learned was discarded.

The decision to limit the size of an example by using only the first 100 words in a message body was driven by efficiency considerations; absent this restriction, the occasional very long message (*e.g.*, Postscript source for a journal paper) would make learning much more expensive. Using the first 100 words in the body of a document is also a sort of feature selection since usually the first few sentences in an e-mail message indicate the message's general content.

Experiments

Preliminary experiments on text

The text version of RIPPER has been tested on a number of categorization problems. Figure 1 gives a flavor for one of these comparisons. This graph summarizes RIPPER's performance, as measured by recall and precision, on a corpus of AP newswire headlines (Lewis and Gale 1994). All statistics are averaged across 10 categories (equally weighted). The various points on the precision-recall curve were generated by varying RIPPER's loss ratio. All training is done on 999-example samples selected by a procedure called "uncertainty sampling" (Lewis and Gale 1994). RIPPER's average performance is comparable or superior to the systems previously applied to this problem, namely C4.5 (Quinlan 1994), a probabilistic Bayesian classifier (Lewis and Gale 1994) and FOIL (Quinlan 1990).[2] I have also obtained similarly encouraging results on the Reuters-22173 corpus (Lewis 1992), another set of news story classification problems (Cohen 1996b).

None of these problems, however, are particularly representative of the categorization problems likely to arise in processing e-mail. One obvious difference is the amount of training data; the examples for the AP titles, for instance, were selected from a total set of over 300,000, far more than could be expected to be available for construction of a personalized e-mail filter. However, there are subtler issues as well. For the AP titles, for instance, the documents are generally much shorter than a typical e-mail message (a little over nine words in length, on average). Also, for both the AP titles and the Reuters-22173 documents, the documents being classified have been written by professionals with the goal of making their subject immediately apparent to a newspaper reader. Finally, the topics, which are generally based on relatively broad semantic categories like "nielsens ratings" or "wheat", are perhaps atypical of the categories of interest in processing e-mail. Elsewhere it has been noted that in the Reuters-22173 data there is considerable variation among topics in the relative performance of different classifiers (Wiener *et al.* 1995); hence it is important to evaluate learning methods on problems that will be representative of the problems actually encountered in practise.

Other researchers (Lang 1995; Armstrong *et al.* 1995) have noted that learning methods based on TF-IDF often perform quite well, even relative to more complex learning learning methods, and my experiments with RIPPER on the AP titles dataset and Reuters-22173 tended to confirm this observation. Further comparisons in this paper will focus on TF-IDF and RIPPER.

Recognizing talk announcements

The category of "talk announcements" was chosen as a more representative test case. A corpus was assembled of all 818 messages posted to a center-wide general-interest electronic bulletin over a 4-year period. All messages were manually labeled as to whether they were talk announcements. I also partitioned the corpus chronologically[3] into a training set of 491 messages and a test set of 325 messages.

TF-IDF and RIPPER were compared on different sized subsets of the training data. The results, averaged over 25 trials, are summarized in the upper left-hand graph of Figure 2. Generally speaking, RIPPER does better than TF-IDF, particularly for small numbers of examples.

The difference is frequently statistically significant. After training on each subsample, a paired test was performed comparing RIPPER and TF-IDF's hypotheses on the test data. In 70 of 175 paired tests, RIPPER's hypothesis was statistically significantly superior, and in only three trials was TF-IDF's hypothesis statistically significantly superior. In the remaining trials the difference was not statistically significant.

[2] Two encodings were used with FOIL, a "propositional" encoding which supports rules similar to the rules constructed by RIPPER, and a "relational" encoding which also supports rules containing some types of phrases. For details see Cohen (1995b).

[3] *I.e.*, every message in the test set was posted after the latest message in the training set.

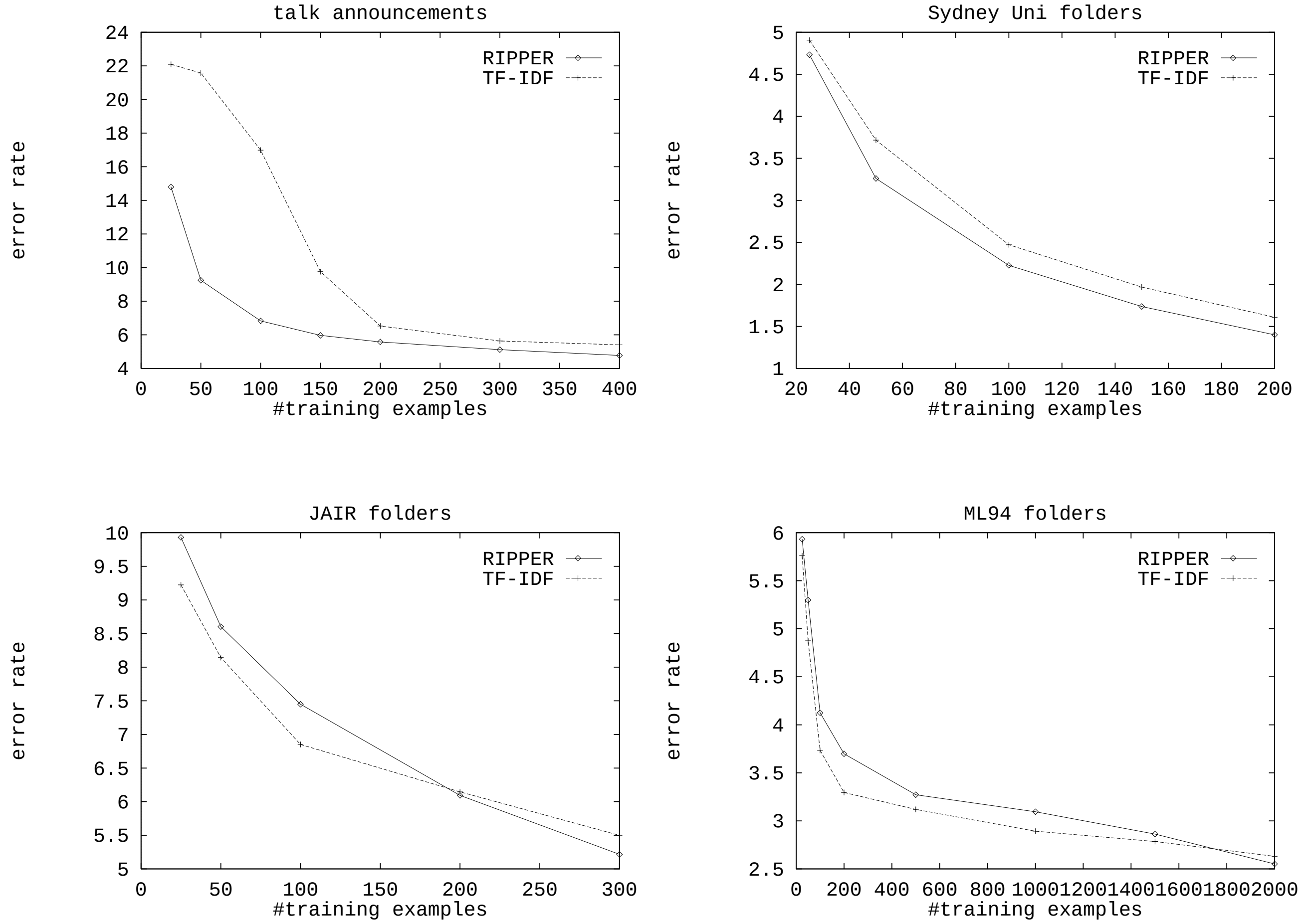

Figure 2: Error rates for various problems

E-mail folders

I also looked at a number of categories which reflect how I file my own mail. My filing system for e-mail messages is currently completely manual and somewhat haphazard, and I am reluctant to use most of the categories of the form "messages to be filed in folder `foo`" as exemplary of useful and/or natural classes. However, some filing subtasks seemed to be reasonable categories to experiment with. I decided to use the following corpora and categories.

Sydney University folders This is a corpus of 269 messages, saved in 38 folders, that were received in September and October of 1995, during a visit to the Basser Department of Computer Science at Sydney University. Many of the folders explicitly correspond to messages from a single sender, or else are highly correlated with the sender. One class was generated for each folder—the messages filed in that folder being positive examples, and the messages filed in other folders being negative examples.

The results, averaged over 25 trials per class and 38 classes, are summarized in the upper right-hand graph of Figure 2. RIPPER performs slightly better than TF-IDF on average, but the comparison is not nearly so one-sided as on the talk announcement problem; it is probably most accurate to say that one these problems the performance of RIPPER and TF-IDF is about the same. In the paired tests, for instance, RIPPER was statistically significantly better than TF-IDF 68 times, statistically significantly worse 52 times, and is statistically indistinguishable on the remainder of the 3013 trials.

JAIR folders This is a corpus of 397 messages, saved in 11 folders, pertaining to my duties as editor of the Journal of AI Research (JAIR). One class was generated for each folder. In contrast to the Sydney University folders, all of these folders are semantically defined. Most of them contain correspondence connected with a single JAIR submission.

The results, averaged over 25 trials per class and 11 classes, are summarized in the lower left-hand graph of Figure 2. Again, the two algorithms are roughly comparable. On average, RIPPER gets a slightly lower error rate with many training examples, and TF-IDF gets a slightly lower error rate with fewer training examples. In the paired tests RIPPER is statistically significantly superior 161 times, statistically significantly worse 184 times, and indistinguishable the remaining 547 times.

ML94 folders This is a corpus of 2162 messages, saved in 71 folders, pertaining to my duties as co-chair of the 1994 Machine Learning Conference. One class was generated for each folder that contained at least 22 messages (*i.e.*, more than 1% of the total), for a total of 21 classes.

These folders are a mixed lot. Most are semantically defined (*e.g.* "budget") but a few are associated with a particular person. Many of the classes are not completely disjoint, and hence a number of messages were filed in multiple folders. The dataset should thus appear noisy to the learning algorithms, which assume the classes to be disjoint.

The results, averaged over 10 trials per class and 21 classes, are summarized in the lower right-hand graph of Figure 2. On these problems the two systems are again comparable in performance, although TF-IDF has a slight edge for most sample sizes. In the paired tests RIPPER is statistically significantly superior 146 times, statistically significantly worse 355 times, and indistinguishable the remaining 993 times. Regardless of statistical significance, however the absolute differences between the two algorithms is small—on average their hypotheses agree on nearly 98% of the test cases.

E-mail filters

In this section I will consider another type of message category—categories useful for filtering or prioritizing unread mail.

The categories I considered were obtained as follows. Recently some colleagues implemented a customizable e-mail reader called Ishmail (Helfman and Isbell 1995). This mail reader allows a user to define an ordered list of *mailboxes*, each of which has an associated classification rule. By convention the last mailbox in the list is called `misc`, and is associated with a classification rule that always succeeds. Before it is read, any incoming e-mail messages is placed in the first mailbox with a rule that "accepts" the message (*i.e.* classifies the message as positive). The classification rules used by Ishmail are boolean combinations of substring and regular expression matches—a language that includes the keyword-spotting rulesets considered above—and, in the current implementation, must be explicitly programmed by the user.

Ishmail allows a number of other properties to be associated with a mailbox. For instance, user can request

User	Problem Name	#positive examples	#negative examples
user1	software	11	256
	conferences	15	256
	talks	87	256
user2	association1	344	3782
	task1	54	3782
	localtalks	81	3782
	subject1	62	3782
user3	subject1	206	1763
	personal	575	1763
	association1	107	1763
	todo	12	1763

Table 1: Summary of the e-mail filtering problems

that the messages placed in a mailbox be automatically archived at set intervals (*e.g.* weekly). A user can also ask to be alerted only when a mailbox has at least k unread messages, or when the oldest unread message is at least d days old; thus Ishmail can be used to prioritize unread mail. One typical use of Ishmail is to put common types of electronic "junk" mail into special mailboxes, which are then given appropriate alerting and archiving policies. Many of these messages—notably messages from specific mailing lists—can be detected reliably by looking only at the sender.

I interviewed a number of Ishmail users and found that several users were employing the system in the following intriguing way. These users defined mailboxes corresponding to particular message categories of interest, but gave these mailboxes either vacuous classification rules, or else highly approximate rules. Messages were then manually moved from the `misc` mailbox (or wherever else they wound up) into the semantically appropriate mailbox. The reason for doing this was usually to take advantage of Ishmail's automatic archiving features on message categories for which it was difficult to write accurate classification rules.

I realized that, in manually correcting missing or erroneous classification rules, and also automatically archiving the results of these actions, these users had been unknowingly providing training data for a learning system. In each case, the messages manually placed into a mailbox are positive examples of the associated classification rule, and messages placed in later mailboxes are negative examples.

It should be noted that all of these users were computer scientists—one a researcher in the area of computational linguistics—and all had written some classification rules by hand. It seems likely, therefore, that these categories are relatively difficult to explicitly program; this makes these categories are particularly interesting as targets for learning.

I assembled eleven datasets of this sort from three different users. The characteristics of these problems are summarized in Table 1. (In the interests of privacy I have replaced the actual user names and mailbox names with rather generic terms.) These datasets represent several months worth of mail, and two of the users had used very complete archiving strategies; hence the datasets are rather large. In each case, I simplified the problem slightly by using only the default "misc" mailbox as a source of negative examples. Two of the eleven categories were sets of talk announcements, providing a nice confirmation of the typicality of the first problem I selected for study.

Figure 3 shows average error rates on these problems. I followed a methodology similar to that used above, training the two learning algorithms on different-sized subsets of the total data, and then measuring the error rates on the remaining data; however, since the sizes of these datasets are different, I used a range of percentages of the whole dataset, rather than a range of absolute sizes for the training set.

Overall RIPPER performs better on these problems. In paired tests, RIPPER is statistically significantly superior 132 times, statistically significantly worse 21 times, and indistinguishable the remaining 331 times.

As an additional (and perhaps more easily interpreted) point of comparison, I also measured the error rates of the two systems on these problems with 10-fold cross validation. The results are summarized in Table 2, with statistically significant differences marked with an asterisk ($*$). RIPPER's error rate is slightly higher on three of the eleven problems, and lower on the remaining eight. On six of these problems RIPPER's error rate is dramatically lower—less than half of TF-IDF's error rate—and RIPPER is never statistically significantly worse.

I will conclude with a few general comments. First, on almost all of problems, RIPPER does somewhat better than TF-IDF, if given a sufficiently large number of training examples. Compared to RIPPER, TF-IDF seems to perform best when there is little training data, and particularly when there are few positive examples. Second, while the performance of the learners with a very small number of training examples is roughly the performance of the default classifier, this performance seems to improve quite rapidly. In the talk announcement problem and the folder problems, for example, there is a noticeable and significant reduction in error with even 100 training examples.

Run-time performance

TF-IDF's run-time performance is relatively well-understood; an efficient implementation requires only a small number of passes over the corpus, leading to linear run-time with a low constant. RIPPER's run-time performance, on the other hand, has been evaluated mostly on very large noisy datasets, and the efficiency of the extension to set-valued attributes has not been previously investigated.

In brief, RIPPER seems to be reasonably efficient for problems of this sort, although perhaps not quite fast enough to be used in an interactive system on current hardware. In learning from the complete set of 491 talk announcement examples, a corpus containing 9760 distinct words and a total of almost 80,000 words, RIPPER requires 43 seconds on a Sun 20/60. About 14 seconds of this time is spent reading in the dataset. (For comparison, TF-IDF takes about 17 seconds on the talk announcement problem, and most of this time is reading in the data.) RIPPER's performance on the 2000-example samples of the ML94 folder concepts is comparable: RIPPER takes an average of 42 seconds to construct a ruleset.

Related work

In a previous related study, the rule learning system SWAP1 was compared to other learned classifiers on the Reuters-22173 dataset (Lewis 1992), a corpus containing 22,173 documents that have been classified into 135 different categories (Apté *et al.* 1994). The documents in the Reuters-22173 collection are all news stories, averaging around 80 words in length, and the training sets used were large—on the order of 10,000 labeled examples.

The focus of this comparison is on text categorization problems that are representative of those that arise in handling e-mail correspondence. As noted above, there is substantial variation even among the different Reuters categories, and little reason to suppose that they would be typical of e-mail categorization problems.

Another technical difference in these two studies is that the text categorization rules learned by Apte *et al.* contained primitive tests that compare word frequency with a fixed threshold, rather than simply checking for the presence or absence of a word. This representation is presumably more accurate but less comprehensible than keyword-spotting rules.

We are currently conducting further studies with the Reuters-22173 collection. One goal of these studies is to compare the performance of RIPPER and SWAP1.

Conclusions

Motivated by an interest in learning classifiers for text that are easy for end users to understand and modify, this paper has compared an extension of the RIPPER rule learning method and TF-IDF on a number of text categorization problems. The benchmark problems used are problems that might plausibly arise in filing and filtering personal e-mail.

One encouraging result of the study is that both methods have fairly steep learning curves; *i.e.*, significant amounts of generalization are obtained with a relatively small number of examples. In each of the graphs in Figure 2, for instance, a significant reduction in the error rate occurs with only 100 examples, and much of the learning takes place with around 200 examples—for many users, only a few days worth of mail. I found this rapid learning rate somewhat surprising—since many of the categories are relatively

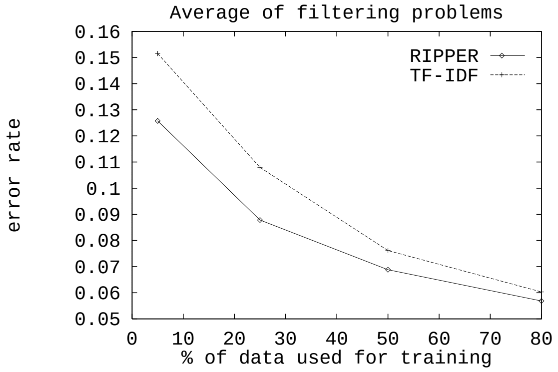

Figure 3: Error rate averaged over the filtering problems

User	Problem	% Error)			
		RIPPER	(se)	TF-IDF	(se)
user1	software	4.53	(1.02)	4.15	(1.28)
	conferences	7.02	(1.48)	5.53	(1.67)
	talks	$*$8.16	(1.37)	20.08	(2.02)
user2	association1	$*$2.84	(0.24)	6.79	(0.38)
	task1	$*$0.65	(0.11)	1.38	(0.22)
	localtalks	$*$0.93	(0.10)	1.89	(0.14)
	subject1	1.48	(0.31)	1.87	(0.26)
user3	subject1	$*$3.60	(0.31)	9.40	(0.78)
	personal	$*$21.00	(0.79)	23.86	(0.69)
	association1	$*$0.91	(0.27)	4.44	(0.45)
	todo	0.73	(0.15)	0.67	(0.15)

Table 2: Performance on e-mail filtering problems (10-CV)

infrequent, one would expect to see only a handful of positive examples in a training set of this size.

Although more work clearly needs to be done, the experiments also shed some light on the relative performance of rule learning methods and traditional IR methods. *A priori*, one might expect rule induction methods to work well when there is a concise keyword-based description of a category, and work poorly in other cases. In filtering e-mail, of course, a number of arguably natural types of categories *do* have simple rulesets. One example is categories that are associated with a single unique sender; for instance, messages from a particular mailing list. Another example is categories distinguished by salient, attention-grabbing keywords; the "talk announcements" category is an instance of this sort of category. (Most talk announcements include a number of salient keywords, such as "talk", "abstract", and "speaker".) However, the experiments of this paper suggest that induction of keyword-spotting rulesets is competitive with traditional IR learning methods on a relatively broad class of text categorization problems. In particular, the rule methods are competitive even in situations in which all or most of the categories are semantically defined.

This suggests that a system which combines user-constructed and learned keyword-spotting rules may indeed be a viable architecture for a personalized e-mail filtering system.

References

Chidanand Apté, Fred Damerau, and Sholom M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

R. Armstrong, D. Frietag, T. Joachims, and T. M. Mitchell. WebWatcher: a learning apprentice for the world wide web. In *Proceedings of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford, CA, 1995. AAAI Press.

William W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference*, Lake Taho, California, 1995. Morgan Kaufmann.

William W. Cohen. Text categorization and relational learning. In *Machine Learning: Proceedings of the*

Twelfth International Conference, Lake Taho, California, 1995. Morgan Kaufmann.

William W. Cohen. Learning with set-valued features. Submitted to AAAI-96, 1996.

William W. Cohen. Some experiments in text categorization using rules. In preparation, 1996.

Jonathan Isaac Helfman and Charles Lee Isbell. Ishmail: Immediate identification of important information. Submitted to CHI-96, 1995.

David J. Ittner, David D. Lewis, and David D. Ahn. Text categorization of low quality images. In *Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, NV, 1995. ISRI; Univ. of Nevada, Las Vegas.

Ken Lang. NewsWeeder: Learning to filter netnews. In *Machine Learning: Proceedings of the Twelfth International Conference*, Lake Taho, California, 1995. Morgan Kaufmann.

David Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning: Proceedings of the Eleventh Annual Conference*, New Brunswick, New Jersey, 1994. Morgan Kaufmann.

David Lewis and William Gale. Training text classifiers by uncertainty sampling. In *Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

David Lewis and Mark Ringuette. A comparison of two learning algorithms for text categorization. In *Symposium on Document Analysis and Information Retrieval*, Las Vegas, Nevada, 1994.

David Lewis. Representation and learning in information retrieval. Technical Report 91-93, Computer Science Dept., University of Massachusetts at Amherst, 1992. PhD Thesis.

J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3), 1990.

J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1994.

J. Rocchio. Relevance feedback information retrieval. In Gerard Salton, editor, *The Smart retrieval system—experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

Gerard Salton. Developments in automatic text retrieval. *Science*, 253:974–980, 1991.

E. Wiener, J. O. Pederson, and A. S. Wiegend. A neural network approach to topic spotting. In *Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, Nevada, 1995.