Identifying SPAM with Predictive Models

Dan Steinberg and Mikhaylo Golovnya

Salford Systems

1 Introduction

The ECML-PKDD 2006 Discovery Challenge posed a topical problem for predictive modelers: how to separate SPAM from non-SPAM email using classic word count descriptions of email messages. The data for the challenge were released around March 1, 2006 and submissions were due June 7, 2006, allowing entrants to devote as much as three months to preparing and modeling the data. We devoted two calendar weeks and three person weeks to this project, the maximum we could spare given other commitments. We found the project appealing for several reasons. First, we started with the belief that modern data mining methods and specifically boosted trees in the form of Jerome Friedman's MART (Multiple Additive Regression Trees) would perform well. Second, our industrial experience at Salford Systems has to date been focused on the analysis of numeric (non-text) data and were eager to gain more experience in the field of text mining. Third, the project organizers had already completed the initial mapping of the text documents to the word count "term vectors" allowing challenge participants to focus on the use of numerical tools and bypass the messy preprocessing of raw text data. Note that the challenge data contained no information regarding the original text; we do not even know the language of the emails let alone the nature of the triggers that could signal SPAM.
The challenge consisted of two tasks, Task A and Task B. As we addressed only Task A we confine our discussion accordingly.

2 Data

We include a brief description of the data here to allow this paper to be self-contained. Additional information may be found in the companion workshop papers and the challenge website http://www.ecmlpkdd2006.org/challenge.html. As the goal was SPAM detection, every email message in the data sets had been tagged as SPAM or not-SPAM (although the class label was not always made available to the modelers). The main training data set consisted of 4,000 email messages evenly divided between the SPAM and not-SPAM classes. The project documents suggest that the SPAM messages were collected from a public "spam trap" (an email address visible only to crawlers and bots, but invisible to humans) but there is no explicit confirmation of this. The not-SPAM emails were also (apparently) collected from "publicly available sources". Confining training data to public sources was a key component of the stated challenge as automatically generated SPAM filters would

ideally not rely on users to manually label their email messages for the purpose of training a SPAM filter. The challenge required participants to develop predictive models that would perform well on individual user email inboxes, and three such inboxes of 2500 unlabeled messages each were provided. Naturally, the distributions of the words used in the public source training data and the individual email boxes are quite different, and this is a major source of the difficulty for machine generated SPAM filters. We have to allow for the fact that individual users vary considerably in their tastes and interests and so what might look like SPAM in the inbox of user #1 might be an explicitly requested message in the inbox of user #2. In addition to the primary data, some "practice" or tuning data was provided. This data consisted of 4000 labeled training emails from public sources and an additional 2500 labeled emails from a single user inbox. The tuning data was supplied with scrambled word indices so that this data could not be pooled with the primary training data. The scrambling ensures that word number 3 in the tuning data does not correspond to word number 3 in the training data, etc. The tuning data could thus be used only to assess the performance of alternative modeling strategies; no specific detail extracted from a tuning data model could be used to improve the models generating the final challenge predictions. It is worth emphasizing that the tuning email inbox was the only example of labeled user data and that the tuning data provided us the only way to learn anything about adapting public source data to individual user SPAM detection.

3 First Stage Data Preparation

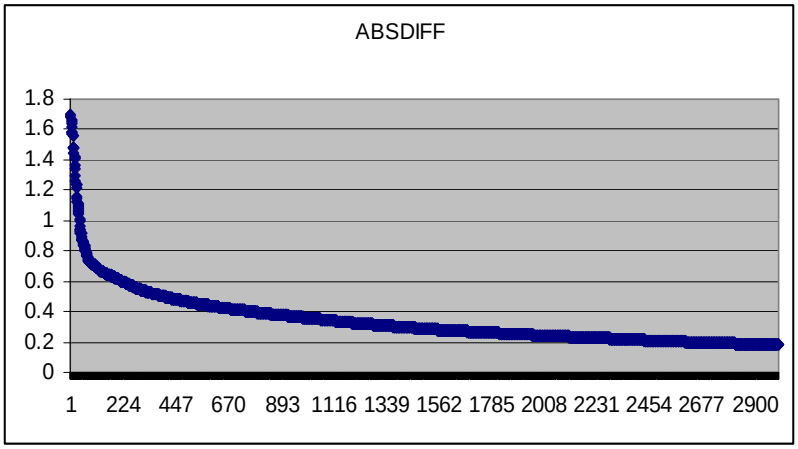
The raw data was delivered in a compressed form often used for sparse matrices. Thus a typical line of data might look like:

```
-1 2:3 9:8 17:3 35:7 71:3 74:2 77:6 85:5 86:1 92:2 94:2 95:6 99:2
```

This record corresponds to not-SPAM as the first element is -1, and it contains 3 occurrences of word #2, 8 occurrences of word #9, ..., and 2 occurrences of word #99. All words in the master vocabulary not listed on the record do not appear in the original email (i.e. they have a frequency or word count of zero). As our analytical and data preparation tools are not equipped to work with such data directly our first task was to expand the data into a complete non-sparse format with zero elements explicitly listed. When the master training data file is so expanded we find a total of more than 150,000 words. The expansion was also performed on all the other data sets. The evaluation data consisted of three user inboxes containing 2500 unlabelled rows of data, with 26580, 27523, and 20227 non-zero word counts respectively. The tuning data consisted of completely labeled data. The public source training data consisted of 4,000 rows and 39967 word counts, and the tuning user inbox consisted of 2,500 rows and 23070 word counts. The next step required combining data from the separate files. For the challenge training and evaluation data one option was to form the union of all words appearing in any file and using this as a master vocabulary. Given the size and cumbersomeness of the resulting files we elected instead to work with intersections. Thus, the intersection of words appearing in both

the training data and a given user inbox formed the master vocabulary for our work on that specific inbox. The number of words in each intersection was 16333, 17791, and 16399 respectively. For the tuning data the word counts were 46219 in the union and 16818 in the intersection of the two files.

Given the large vocabularies some form of attribute selection was advisable and we used a simple statistical test popular in bioinformatics. We first limited the vocabularies separately for each inbox, selecting the subset of words found in both the inbox and the training data. For each attribute so selected we calculated a modified t-test due to Tibshirani et. al (2001) for the difference of means between the SPAM and not-SPAM classes in the training data and then ranked the attributes by the absolute value of this t-statistic. The tests were organized separately for each user inbox and restricted to the vocabulary common to the train data and the inbox in question. An example graph of results appears below for the first user inbox. It plots the statistics against the rank order of the attribute.



This is not our preferred approach to attribute selection as it is based on a myopic univariate discriminant analysis but it was easiest to deploy. The t-tests rank the predictors and it was up to us to select a cut-off for attribute selection. We experimented with several different cutoffs in our initial modeling, trying as few as 1,000 and as many as 8,000 attributes to arrive at a preferred total of 3,000 predictors. All further data preparation and model generation reported below was based on an inbox-specific selection of the top 3,000 attributes.

4 Second Stage Data Preparation

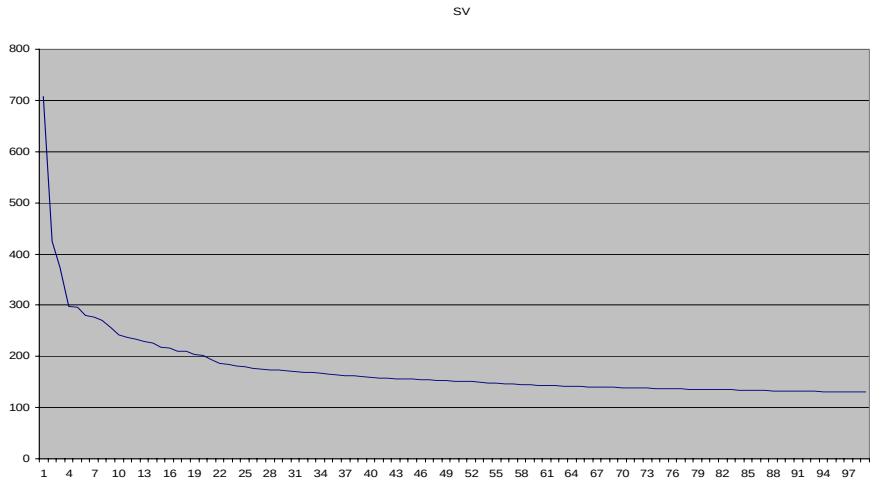
As email messages can vary greatly in length it is difficult to compare or cluster records based on the raw word frequencies. To adjust for the email length we converted the counts to relative frequencies by dividing each term count by the email total word count (so that the sum of relative frequencies is 1 for all records).

Following the text mining literature we also added the following summary document features describing each email:

- LEN: document size, total number of words,
- NNZ: number of unique words (Number NonZero),
- MAXF: count of most often used word,
- ENTR: document entropy, calculated over words,
- KL: distance to the target class training data centroids (Kullback-Leibler statistic)

Each feature was constructed in two versions: on the original document vocabulary (version 1) and on the restricted vocabulary found in the intersection between the training data and the specific inbox (version 2 or version R).

Previous research (Dumais et. al. 1988) has suggested that some form of data compression or summarization is vital for the analysis of word count data and we decided to follow this practice, compressing the data with a Singular Value Decomposition (SVD). The transform was applied to the subset of the best 3,000 relative word counts discussed above. The SVD is similar to Principal Components Analysis (PCA); it creates linear combinations of the original data (whether in raw count or relative frequency form) and effectively captures information on joint patterns of occurrences of individual words. The SVD transformation allows us to capture some types of relationships between words which is precisely what is missing in the raw word count data. One should expect such transformed data to yield better predictive performance than the raw (or normalized) counts. The SVD transformation usually allows a radical reduction in the number of predictors required as much of the information content of the raw data is captured in the leading SVD vectors. We experimented with keeping different numbers of SVD vectors and settled on 20 for most of our modeling work. (We have heard text mining practitioners recommend 40 as a rule of thumb). The screen plot below of the first 100 singular vectors plots the Singular Value against the singular vector number is representative of the results we obtained separately for each user inbox.



Note that the SVD transformation is a second stage of data reduction and follows the first stage of raw attribute selection. The SVD decomposition makes no reference to the target; it is based only on the predictors and can be generated from the training data or from the pooled training and unlabeled user inbox data. Below we report that restricting the SVD to the training data alone yields slightly better results. The SVD can also be computed with or without the summary document features listed in the table above, and the summary document features can also appear as separate predictors in their own right. We experimented to identify the best combination of predictors and inputs into the SVD.

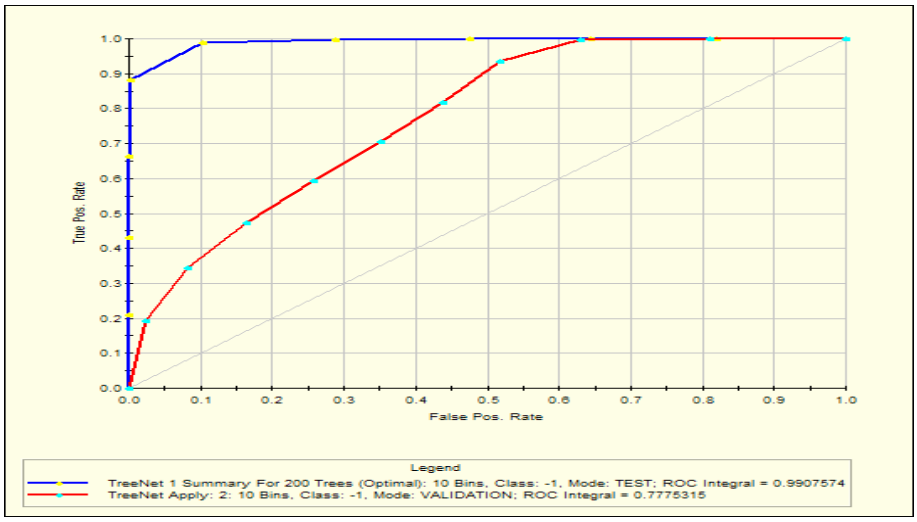
5 General Modeling Methodology

We elected to use TreeNet(tm), the commercial release of Friedman's stochastic gradient boosting as it has performed well in a broad range of real world predictive modeling challenges. Early exploratory runs were based on small ensembles of 200 trees constructed with a moderate learning rate of 0.10 to reduce run times. Our final models were run with a slow learning rate of 0.01, using the binary logistic loss function, and growing 1000 6-node trees. Parameter settings were chosen via experiments on the tuning data described below.

6 Modeling Details

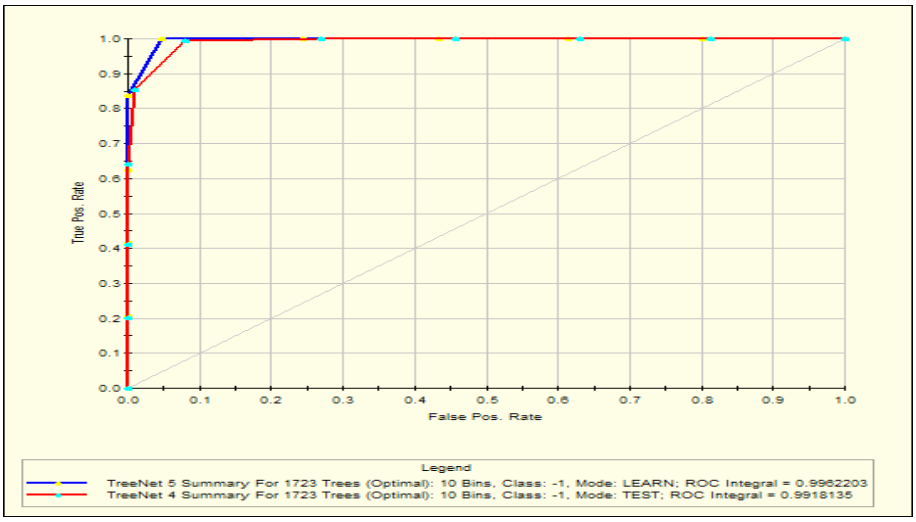
The results we report here are based on what was known to us at the time of the challenge, namely the tuning data. A revised analysis based on the labeled evaluation data would be a desirable undertaking but is not included here. We began with a simple experiment to determine how well a model built entirely on the public source

data would perform on an individual inbox. We divided the training portion of the Tune sample into random halves and built a quick TreeNet model using only the top 3000 ranked raw word counts as predictors and then used the "optimal" model to score the data from the Tune sample inbox. The graph below displays the ROC curves for the two test samples.



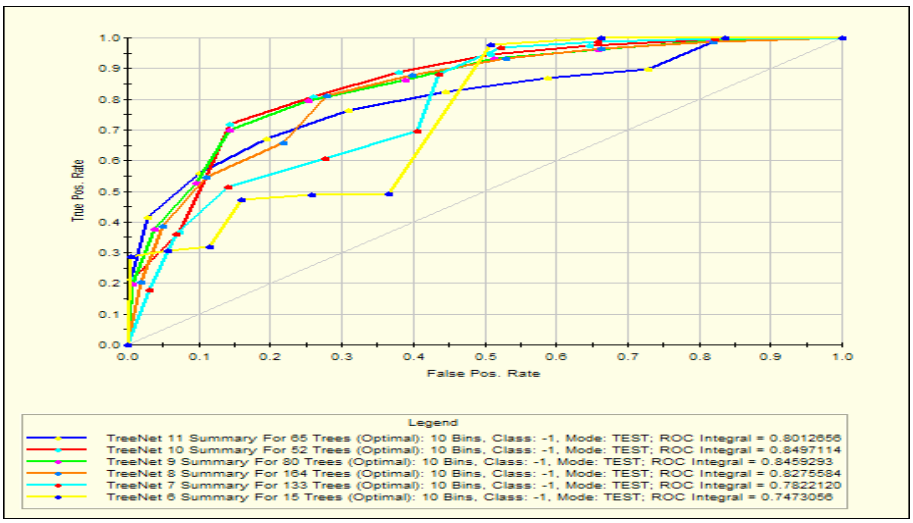
Note that the performance on public source data (the upper curve) is excellent. Even a crude 200-tree ensemble reaches an area under the ROC curve (AUC) of .991 on test data. But when the same model is applied to the rather different distribution of emails and vocabulary found in a real individual's inbox the AUC drops dramatically to .778. While .778 is a value that could be welcomed with enthusiasm in some applications it is far too low for SPAM detection and most email users would find the corresponding classification error rates of such a model annoying.

If we take the 2500 email individual user inbox tuning data, randomly set aside 1/3 of the data for test and develop a model on this individual user's data we again obtain excellent results with an AUC on test data better than .99: In the graph below the two ROC curves correspond to the training and test portions of the individual user's data. This tells us that successful inbox-specific models can be developed using naïve raw word counts as predictors. But the model is completely customized to this user's inbox.



The Challenge requires us to build predictive models from data that are somewhat less relevant than can be found in an individual's inbox, and we turn now to the models built in this way.

The models we report here are all based on the Tune data and were used to guide the parameter settings we used in the final challenge submission. The graph below displays the ROC curves and AUC values achieved on the single user tune inbox.



The lowest ROC curve (over the lower half of the graph) corresponds to the naïve raw word count model. The better models are based on various combinations of SVDs as predictors and document summary features. The best results were obtained when the SVD vectors were based only on the training data instances, but with the document

summary features included in the SVD construction, and with the document summary features also included as separate predictors. These results are very similar to those we obtained on the evaluation data.

7 Comments on Results

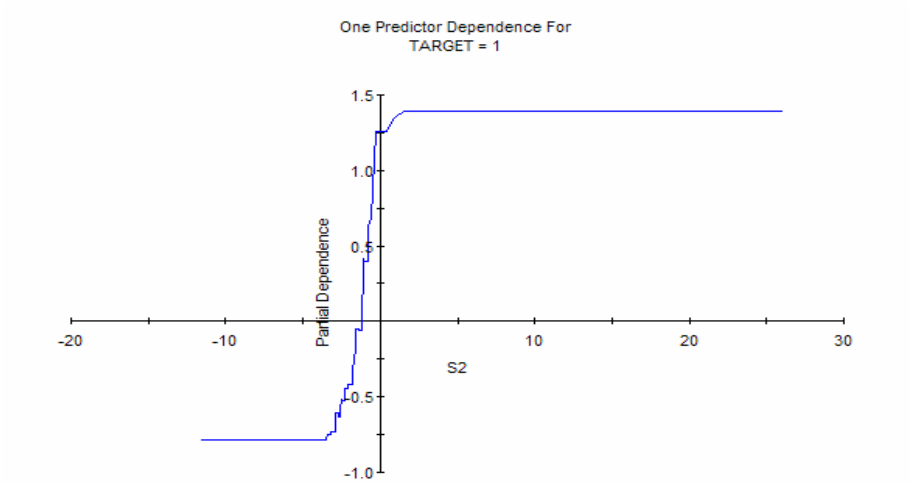
As the details of the data are hidden from us we cannot discuss specific indicators predicting SPAM other than the meaningful features we constructed from the abstract data. We can extract some useful but limited insights and we present those here. Based on the TreeNet variable importance ranking we see that the 2nd SVD vector is ranked most important, and that top 4 variables are all SVD vectors. The document summary features do play a role however, with the 3 such features appearing in the top 10 predictors.

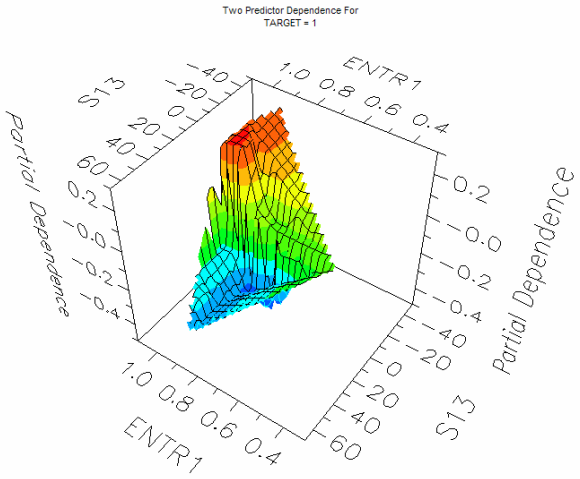
TreeNet Variable Importance

Variable	Score	
S2	100.00	|||||||||||||||||||||||||||||||||||||||||||
S9	49.11	||||||||||||||||||
S3	33.89	||||||||||||
S5	33.27	||||||||||||
ENTR1	25.53	|||||||||
S13	22.85	|||||||||
S4	19.85	||||||||
S29	19.30	|||||||
KL2P	18.64	|||||||
NNZ1	16.46	||||||
S12	15.52	||||||
S21	13.51	|||||
S27	12.83	|||||
S25	12.03	||||
LEN1	11.89	||||
S10	10.81	||||
S17	9.79	|||
S23	9.61	|||
S1	8.58	|||
S26	8.30	|||
MAXF2	8.22	|||
S14	8.04	||
S28	7.26	||
KL2M	7.16	||
S11	7.08	||
S16	6.96	||
S18	6.04	||
S24	5.87	||
S8	5.37		
S20	5.25		
S22	4.14		
S6	4.13		
S15	2.46	
NNZR	2.41	
S19	0.00	
S7	0.00	

ENTR2	0.00	
MAXF1	0.00	
LENR	0.00	
S30	0.00	

The TreeNet dependency plots for the top variables are almost all sigmoids or ramp functions, similar to that shown below for the most important predictor S2 (of course some have a negative effect and are thus downward sloping from left to right). There are evidently powerful interactions captured in the model as illustrated in the 3D graph below showing the dependency of the target on ENTR1 and S13. :





8 Improving the Results

During the challenge we conjectured that there were at least two additional ways to improve the models. First, our list of features is rather small and others could be created, in particular features based on the prevalence of words appearing in each target class. Second, we suspect that reweighting the training data to make it more similar in word distribution to any individual inbox might also be helpful.

References

1. Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and Regression Trees. Wadsworth & Brooks/Cole, California.
2. Dumais S. T. , and G. W. Furnas and T. K. Landauer and S. Deerwester and R. Harshman (1988). Using Latent Semantic Analysis To Improve Access To Textual Information. CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems, 281—285, Washington, D.C.
3. Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29:1189–1232.
4. Tibshirani, Robert, Trevor Hastie, Balasubramanian Narasimhan, Michael Eisen, Gavin Sherlock, Pat Brown, and David Botstein. (2001). Exploratory screening of genes and clusters from microarray experiments. http://www-stat.stanford.edu/~tibs/ftp/samclus.pdf.