Spam or Not Spam – That is the question[*]

Ravi Kiran S S and **Indriyati Atmosukarto**
{kiran,indria}@cs.washington.edu

Abstract

Unsolicited commercial email, commonly known as spam has been known to pollute the medium of electronic mail. Recipients of spam messages must waste time deleting annoying and possibly offensive messages. One problem that can arise is the possibility of users deleting legitimate messages. This paper presents analysis on various aspects of robust spam filtering. Experiments were performed to compare the performance of various learning classifier for identifying spam messages. Additional experiments were carried out to reveal the importance of choosing statistically meaningful attributes to in improving the performance of spam classifiers. We also compared the performance of ensemble learning classifiers against single classifiers. Lastly, we investigated the 'temporal evolution' of spam messages over a period of time. We conclude by providing some pointers to future research work for developing robust spam classifiers.

1 Introduction

The Merriam-Webster Online Dictionary (`http://www.m-w.com`) defines spam as "unsolicited, usually commercial, email sent to a large number of recipients". Given the increasing dependence of today's society on Internet, spam has become an electronic thorn in the foot of the ubiquitous computer user. Spam takes away resources from users and service suppliers without compensation or authorization. Spam emails are typically sent using bulk mailers and address lists obtained from web pages and newsgroup archives. Their content varies from deal to real estate to pornography.

Unfortunately, the definition of spam is subjective: one person's spam could easily be another person's gold. Hence, spam filters have a unique challenge in ensuring effectiveness and accuracy. Effectiveness is measured by the percentage of spam caught. This percentage should be as high as possible. Accuracy is measured by the percentage of emails incorrectly classified as spam. This second percentage should be as low as possible.

In order to solve to spam filtering problem, researchers have directed efforts on numerous statistical and non-statistical approaches to design *spam classifiers*. The increasing relevance of research in this topic is evident from the proceedings of two recent conferences, (CEAS-2004) and (Spam 2004). Both conferences aim to bring together communities from diverse fields such as machine learning, cryptography, natural language processing, security, networks, systems and human computer interaction with the common goal of eliminating spam messages.

This paper is organized as follows: Section 2 provides a discussion of various spam filtering approaches and describes state-of-the-art techniques used for the purpose. In Section 3, we provide a performance evaluation of various machine learning algorithms such as Decision Trees, Naive Bayes, Support Vector Machines, and ensemble techniques such as Ensemble Decision Trees, Stacking and Boosting. We also analyze the importance of statistically meaningful attribute selection and provide some interesting results for analyzing the temporal evolution of spam. Section 4 concludes with some pointers to future research directions for developing robust spam classifiers.

2 Existing Approaches: A survey

A study of the ever increasing research on spam filtering shows that methods fall primarily under two categories – statistical and non-statistical (CEAS-2004). We examine the non-statistical approaches first.

One category of such approaches use the service provided by email server to attack spam messages at the network level. Using these approaches, email servers can now block spam messages based on the Real Time Black Hole list (RBLs) (Popular RBL services are MAPS(`http://mail-abuse.org/rbl`) and SpamCop (`http://www.spamcop.net`)). RBLs are an at-

[*]Loosely inspired by William Shakespeare's *Hamlet*, `Act III,Scene 1.`

tempt to maintain lists of hosts (mail servers) that are used regularly by spammers. When a mail server receives a connection from another server trying to send a message, it can check against the list whether the connecting server is in the RBL list. If it is, the mail server can choose to refuse to accept any mails. This approach is not the most effective method as the chance of blocking a legitimate message is quite high. One spam sender can result in hundreds of legitimate users being blocked by the list. Another problem with the earlier approach is that it focuses only on the content in the email header.

This motivated algorithms to filter spam based on *keywords*. Essentially, they employ hand-crafted rules to search for particular keywords, phrases, or suspicious patterns in the message (SpamAssassin), (Sahami *et al.* 1998), (Schleimer, Wilkerson, & Aiken 2003).

An interesting example in the non-statistical category is Vipul's Razor (Vipul). The approach employs collaborative filtering while relying on a network of servers which store spam message signatures to verify if an incoming message is spam. If the incoming message escapes detection, the user can report it so that all the other users can avoid it (by virtue of using the same network of servers).

Another approach that is complementary to maintaining black-lists is maintaining white-lists, where the spam filter sends mails to senders not in the user's white-list, asking them to answer a simple question in order to rule out spamming robots.

The problem with the previous two approaches and non-statistical approaches is that *there is no learning component* to admit messages whose content 'looks' legitimate. This may lead to undetected spam and in the latter case, a frustrating proliferation of automatic answer-seeking replies.

Due to the above limitations, developers of anti-spam filters are increasingly drawn to the potential of statistical machine learning. Consequently, there is a growing number of spam filter implementations that incorporate learning-based components. With its roots in text classification research, learning-based spam filtering seeks to answer the question "Is message x spam?". How this question is answered varies upon the type of classification algorithm in place. But while categorization method differs between statistical filters, their basic functionality is similar. The classification algorithm uses feature vectors to judge the documents.

One of the widely used methods, Bayesian classification, attempts to calculate the probability that a message is spam based upon previous feature frequencies in spam and ham (Androutsopoulos *et al.* 2000), (Sahami *et al.* 1998), (Graham 2003). A notable example is the open source software `SpamBayes`[1]. The popularity of Support Vector Machines has been enhanced with successful applications in spam classification (Drucker, Wu, & Vapnik 1999), (CEAS-2004). Other popular learning algorithms applied to spam filtering include boosting (Carreras & Márquez 2001), and neural networks.

3 Experiments

In this section, we describe experiments performed in order to analyze various design and implementation aspects of spam filtering. Our focus was on the classification algorithms that could be used for the purpose. In particular, we looked at the following:

1. Decision Trees (DT)
2. Support Vector Machines (SVM)
3. Naive Bayes (NB)
4. Neural Networks
5. Ensemble Decision Trees (EDT)
6. Boosting
7. Bagging
8. Stacking

For a brief but very readable description of these methods, the reader is encouraged to consult (Mitchell 1997) and (Russell & Norvig 2003). In addition, we also examined the temporal evolution of spam and explored heuristics that could be used to improve the performance of one of the most widely used classifiers – Naive Bayes[2].

For all the experiments, we used the open source WEKA framework (Witten & Frank 2000) to test the classifiers. WEKA is a collection of machine learning algorithms for data mining tasks. It is well-suited for developing new machine learning schemes. Our emphasis has been on the performance analysis of the aforementioned algorithms so that the results assist in developing novel and effective spam filters. Moreover, WEKA has a highly customizable interface and efficient implementations which enabled us to run a large number of experiments. For this reason, we have used WEKA rather than get bogged down by implementation details and tweaks thereof for various classification methods.

For all the experiments, we used a training set of labeled spam/ham mails to train the classifiers. Testing was performed on a testing set, unseen by the classifier and the performance was measured by evaluating the accuracy, precision and recall for various techniques. Each of the experiments is described in detail below.

[1]`http://spambase.org`

[2]Ironically, Naive Bayes has also been abused for a variety of reasons which we point out later.

Experiment 1 - Evaluating various classifiers on `spambase` dataset.

For our preliminary experiments, we used the `spambase` corpus (Forman 1999). The dataset contains 4601 instances of email, of which about 39% is spam. Each instance corresponds to a single email, and is represented with 57 attributes plus a class label (1 if spam, 0 if not). The data files contain one instance per line, and each line has 58 comma delimited attributes, ending with the class label. Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail; frequency is encoded as a percentage in $[0, 1]$. A few attributes measure the length of sequences of consecutive capital letters. The dataset and the associated documentation can be accessed publicly at (`spambaseftp`).

The performance of 8 different algorithms is presented as a series of ROC curves in Figure 1. In the current context, the value of (1 - Precision) tells us how often a classifier will label a legitimate email(ham) as spam. Recall tells us how effective the classifier is at detecting spam. Although Nearest Neighbor was not included in the original list of classifiers, we have shown it in order to provide a contrast and also to rule out methods which employ simplistic distance-based measures[3]. Also note that the ROC curve for Nearest Neighbor indicates that higher recall comes at an unacceptable price – an increase in the likelihood of ham being classified as spam. This observation merely provides support to a fact well known in the machine learning community —- Nearest Neighbor classifier is a poor generalizer. Naive Bayes and Decision Trees afford extremely good accuracy because of the inherent simplicity of their models but since this simplicity leads to search in a restricted hypothesis space, they seem to perform not as well as Neural Networks and Support Vector Machines. Note however that the hypothesis restriction is somewhat relaxed when ensemble methods are employed, leading to an improvement in performance . As the curves demonstrate, ADABOOST and Ensemble Decision Trees provide the best performance in the operational region. The significance of this result is explored separately in Experiment 3. Table 1 contains the accuracy each algorithm attains on the test set. The results are consistent with the Precision-Recall curves of Figure 1.

We performed additional experiments on the `spambase` dataset by performing a random $50\% - 50\%$ split into ham /spam for training data and a leave-one-out cross-validation and k-fold cross validation where we chose $k = 10$. However, we did not observe any significant improvements from above results.

The `spambase` corpus serves as a good testbed for evaluating a new spam filter. However, it has certain limitations. The first among them is that the attributes were selected on the basis of email arriving at one particular individual at one particular corporate organization[4] and hence, not representative of a general spam sample. Extracting these attributes from other email collections could result in rather sparse data. Figure 2(left) shows that the occurrences of each attribute for a different dataset (`SpamAssassin`) are infrequent. This indicates the unsuitability of `spambase` attributes for developing spam classifiers.

Table 1: Classifier accuracy performance on `spambase` dataset. The accuracies were sorted in decreasing order. Ensemble Decision Tree had the highest accuracy, while the Nearest Neighbor classifier performed the worst.

Classifier	Accuracy
Ensemble Decision Tree (# of tree = 25)	96.40
Adaboost	95.00
Stacking	93.80
SVM	93.40
Bagging	92.80
Decision Tree	92.58
Neural Network	90.80
Naive Bayes	89.57
Nearest Neighbor (k = 5)	89.40

Experiment 2 - Importance of meaningful attributes for classification.

Due to the above limitation of the `spambase` dataset, the second experiment was performed on a bigger and more general dataset: the `SpamAssassin` corpus (SpamAssassin). The dataset contains 8681 instances of email, of which 2230 are spam. We divided the dataset into 5791 training instances (1490 spam and 4301 ham) and 2890 testing instances (740 spam and 2150 ham). From this dataset, we extract attributes based on the well known tf-idf weighting (Jones 1972). We used `ifile`[5] preprocessor to extract the list of words in email[6]. Each instance was then represented as a feature vector of 100 words obtained using the `tf-idf` weighting method plus one class label.

For comparison, we extracted `spambase` feature attributes from the `SpamAssassin` dataset. The attribute

[3]Note that k-NN uses a Euclidean or Mahalanobis distance measure.

[4]George Forman, an employee at Hewlett-Packard.

[5]`http://www.nognu.org/ifile`

[6]More conveniently, `ifile` removes frequently used stop words.

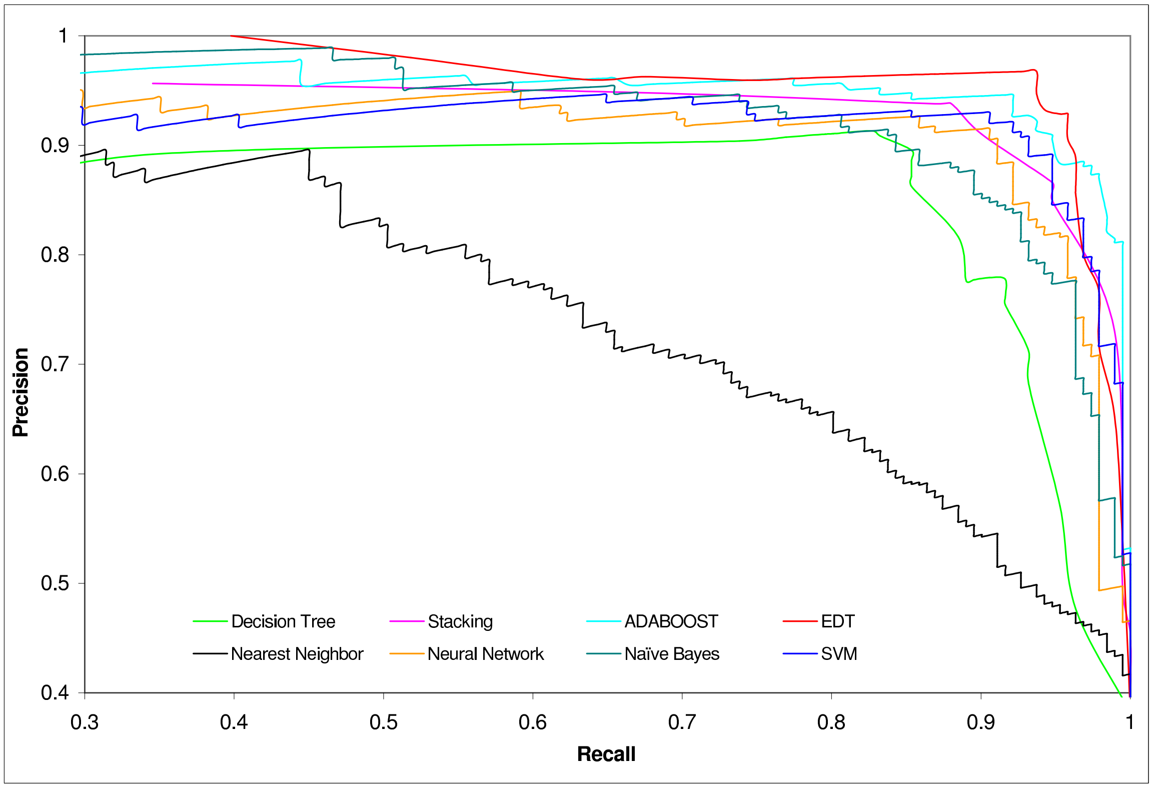

Figure 1: ROC curve of the 9 classifiers on spambase test data. Note that ADABOOST and EDT perform the best, closely followed by other ensemble methods. SVM and Neural Networks perform competitively as they can model complex hypotheses. The simplifying assumptions of Naive Bayes and Decision Trees cause them to perform slightly poorer. As expected, Nearest Neighbor performs the worst.

presence is much more pronounced for tf-idf attributes (see Figure 2 for a comparison with `spambase` attributes). The superiority of tf-idf attributes is evident from the ROC curves in Figure 3, which show that classifiers perform better when tf-idf attributes are used. This result emphasizes the importance of extracting meaningful attributes for improving classification performance. The accuracies achieved by each attribute type shown in Table 2 support this observation.

Experiment 3 - Ensemble vs. non-ensemble classification.

A combination of multiple classifiers may be able to produce an overall classifier which is more stable and accurate than any of its components. This observation, coupled with the fact that ensemble classifiers tend to be more robust to over fitting while avoiding the instability problems of local minima, has made ensemble classification the darling of machine learning community. Prompted by the success of methods such as Bagging (Quinlan 1996) and Boosting (Schapire 2002), we employed them to learn ensemble classifiers – ADABOOST, Stacking and Ensemble Decision Trees. We compared the best performances of three different ensemble classifiers and three single classifiers. For EDTs, we tried different number of trees to represent the ensemble and found that 25 trees resulted in the best performance. An ensemble of Decision Tree, SVM and Naive Bayes with ADABOOST as the meta-learner proved to be the best classifier combination for Stacking. ADABOOST performed particularly well in boosting the performance of weak classifiers (see Figures 1 and 3). Our results (shown in Table 3) are consistent with those reported in machine learning literature – ensemble methods generally perform better than single classifiers.

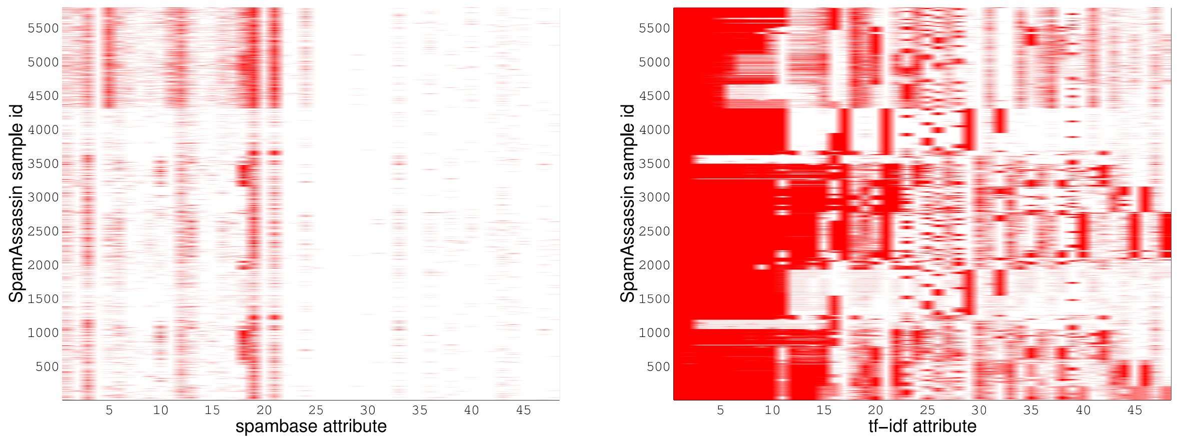

Figure 2: A graphical representation of attribute presence in SpamAssassin dataset for spambase (left) and tf-idf (right). Note the sparsity of attributes for spambase in the left figure. For each attribute type, we show 48 significant word-level attributes.

Experiment 4 - Tackling the poor assumptions of Naive Bayes.

Judging from the number of spam filtering software which use Naive Bayes as the classification engine, it would not at all be an exaggeration to say that Naive Bayes is the 'workhorse of spam classification'. This is a consequence of its attractive features - simple model, easy to implement and speed . However, Naive Bayes has its own share of shortcomings, primarily arising out of its severe assumptions and in certain contexts, it has been denigrated as the "punching bag" of classifiers. (Rennie *et al.* 2003) analyzed the multinomial Naive Bayes model and proposed simple, heuristic solutions in an attempt to tackle its poor assumptions. However, they presented their results for text categorization. Naturally, it would be interesting to observe the performance on spam filtering – which is after all, a form of text categorization. For this, we used the `ComplementNaiveBayes` module in WEKA. As the results in Table 4 demonstrate, the algorithm performs better than the original Naive Bayes model and is competitive with other classification algorithms.

Experiment 5 - Tracking the temporal evolution of spam.

A recently observed trend is that spammer techniques have evolved in response to the appearance of more and better filters. As soon as companies develop effective filters, spammers change their tactics to avoid the new spam blockers. This leads to a vicious cycle, with spammers actually reinvesting their profits into developing new techniques to evade new spam filters. During the course of our literature survey, we noticed that most of the existing evaluations of spam detection algorithms completely ignore this 'evolution-in-time' issue, randomly selecting training and testing sets from a email corpus. If this temporal evolution can be learned, spam filters can adapt to the ever-changing tactics of spammers with greater ease. As a first step towards the development of this genre of adaptive spam filters, we performed an analysis of attribute-level changes in spam over time. For this, we extracted all email instances in year 2002 from the `SpamAssassin` dataset and split the data into 6 different time periods: January - August, January - September, January - October, January - November, and January - December 2002. We then trained these 6 datasets using the $J48$ Decision Tree classifier. $J48$ selects the most prominent attributes based on the maximum Information Gain criterion (Cover & Thomas 1991). In Table 5, we list 7 most prominent attributes seen for each of the 6 time periods described previously. The results show that there is some evolution in the spam data from August to September, but not in the subsequent months. A closer observation reveals that many of the selected attributes are HTML keywords such as `td` and `font`. This is because spammers typically send mails in HTML format. Also notice the presence of a familiar spam keyword `click` which persists over time.

4 Conclusions and Future Work

In this paper, we analyzed various aspects of spam filtering. We focused on the performance evaluation of various machine learning algorithms for the task of detecting spam by performing various experiments. As an initial experiment, we compared 9 classifiers on `spambase` dataset using `spambase-attributes`. This experiment was helpful in determining the best classifiers, however, further analysis revealed that the attributes were very specific to the

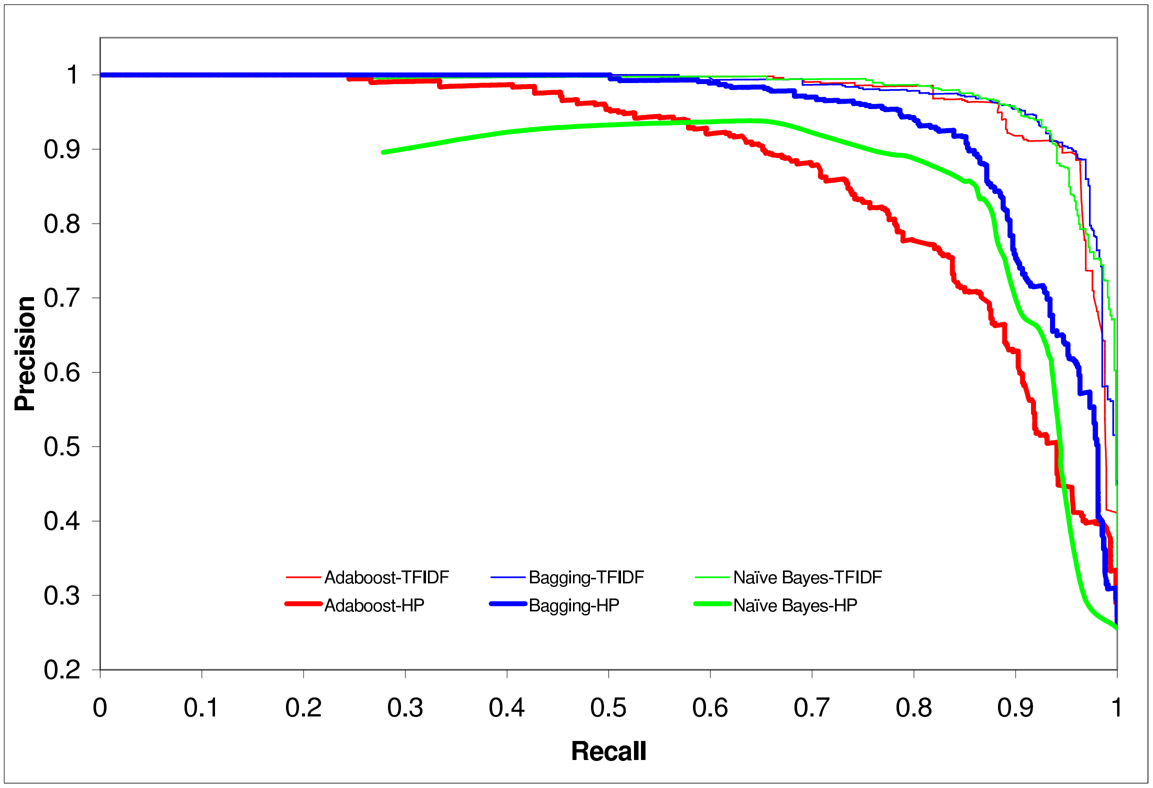

Figure 3: ROC curve of the three best classifiers on `SpamAssassin` dataset. The curves show that tf-idf attributes (thin lines) provide better performance compared to `spambase` attributes(thick lines).

Table 2: Comparison of the best three classifier on SpamAssassin dataset represented using `spambase` attributes and tf-idf attributes. tf-idf attributes gives better accuracy performance for all three classifiers.

Classifier	Accuracy	
	`spambase` attributes	tf-idf attributes
Bagging	94.12	96.06
Naive Bayes	86.82	96.23
Adaboost	89.62	96.06

Table 3: Performance comparison – Ensemble classifiers vs single classifiers. DT = Decision Tree, NB = Naive Bayes, NNet = Neural Network, EDT = Ensemble Decision Tree, AB = Adaboost, Stk = Stacking.

	DT	NB	NNet	EDT	AB	Stk
Prec	90.70	89.80	89.70	96.80	94.60	93.90
Rec	90.40	83.20	85.00	93.70	92.10	88.00
Acc	92.58	89.57	90.80	96.40	95.00	93.80

dataset and not representative of general spam.

To alleviate this, we experimented with various attribute selection methods and observed that attributes that are statistically meaningful have a dense presence over the entire data (spam/ham). An attractive feature of such attributes is that they tend to be data-independent and can provide better separability over many kinds of spam/ham. In particular, we employed the *tf-idf* (Jones 1972) weighting method as a statistically meaningful method for extracting attributes and found that it improved the performance of the classifiers.

Prompted by the success of ensemble learning methods, we built ensemble classifiers – ADABOOST, Stacking and Ensemble Decision Tree and compared their performance with the best performances of single classifiers. Our results confirm findings in literature – ensemble classifiers entail

Table 5: Temporal evolution of spam – The seven most important attributes for each month during the period August 2002 - December 2002. The attributes are chosen by a breadth first traversal of the corresponding decision tree.

Attribute	Aug	Sep	Oct	Nov	Dec
First	precedence	td	td	td	td
Second	references	date	date	date	date
Third	ist	click	click	click	click
Fourth	font	references	references	references	references
Fifth	exim	font	work	work	work
Sixth	received	linux	font	font	font
Seventh	td	imap	linux	linux	link

Table 4: Comparison between Complement Naive Bayes and original Naive Bayes on `spambase` dataset.

Classifier	Precision	Recall	Accuracy
Naive Bayes	0.79	0.691	81.2
Complement Naive Bayes	0.775	0.723	81.4

superior performance compared to single classifiers.

Following (Rennie *et al.* 2003), we experimented with the multinomial Naive Bayes model and extended their results. We showed that tackling the poor assumptions present in the Naive Bayes has the potential to increase its popularity in the spam filtering community.

Spammers change tactics over time and this is reflected as a 'temporal evolution' of spam attributes. As an initial study of this 'evolution', we analyzed attribute-level changes in spam messages over time and found that spammers tend to send HTML formatted messages. It would be interesting to observe this evolution for a longer time period and use adaptive prediction techniques from machine learning. An interesting work in the same flavor is presented in (Dalvi *et al.* 2004) which treats spam detection as a two player adversarial game between the spam filter and the spammer. We believe that this paper is the first step towards a potentially rich area for future research. We defer a discussion on this for future work.

Among other directions for future research, we contemplate using a Mixture of Naive Bayes model (Nigam *et al.* 2000) as an alternative to Naive Bayes model. Also, to handle common morphological and inflectional endings, the word stemming algorithm (Porter 1980) can be used. For instance, this converts *taking* to *take*,*cardiovascular* and *cardiology* to *cardio* etc. so that variations of the same spam keyword can be detected more effectively. Another interesting direction would be to apply Latent Semantic Indexing (Papadimitriou *et al.* 1998) as a means of learning relevance-driven spam classifiers.

To conclude, the problem of spam classification has several aspects to it, notably in the design of robust classifiers. However, emerging trends suggest that this problem is more likely to be solved by a combination of several techniques involving non-statistical (hand-crafted rules, white-lists, black-list, Vipul's Razor) and statistical (machine learning) techniques.

5 Acknowledgements

We would like to thank Dan Weld for his comments on the first project and various suggestions during the project meetings, particularly for suggesting that we explore the temporal evolution of spam.

We would also like to acknowledge the developers of WEKA for providing an efficient testbed for machine learning algorithms and Larry Wall for developing the Swiss army knife of scripting – *perl*.

A Division of Work

Both of us worked jointly on the following aspects of the project:

1. Literature Survey.
2. Data collection.
3. Preprocessing scripts. (written in perl)
4. Evaluation of various classification methods.
5. Report draft.

References

Androutsopoulos, I.; Koutsias, J.; Chandrinos, K.; Paliouras, G.; and Spyropoulos, C. 2000. An evaluation of naive bayesian anti-spam filtering.

Carreras, X., and Márquez, L. 2001. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing*.

CEAS-2004. http://www.ceas.cc.

Cover, T. M., and Thomas, J. 1991. *The Elements of Information Theory*. John Wiley.

Dalvi, N.; Domingos, P.; Mausam; Sanghai, S.; and Verma, D. 2004. Adversarial classification. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, 99–108. ACM Press.

Drucker, H.; Wu, D.; and Vapnik, V. 1999. Support vector machines for spam categorization. *IEEE Transcations on Neural Networks* 10.

Forman, G. 1999. //http://www.ics.uci.edu/ mlearn/mlrepository.html.

Graham, P. 2003. Better bayesian filtering. In *Proceedings of the First Annual Spam Conference*. MIT Press.

Jones, K. S. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28:11–21.

Mitchell, T. 1997. *Machine Learning*. McGraw Hill.

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. M. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39(2/3):103–134.

Papadimitriou, C. H.; Tamaki, H.; Raghavan, P.; and Vempala, S. 1998. Latent semantic indexing: A probabilistic analysis. 159–168.

Porter, M. 1980. //http://www.tartarus.org/martin/porterstemmer/.

Quinlan, J. 1996. Bagging, boosting and C4.5. In *Thirteenth National Conference on Artificial Intelligence*. AAAI/MIT Press.

Rennie, J.; Lawrence, S.; Teevan, J.; and Karger, D. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of ICML*.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence - A Mordern Approach*. Prentice Hall.

Sahami, M.; Dumais, S.; Heckerman, D.; and Horvitz, E. 1998. A bayesian approach to filtering junk email. In *Proceedings of AAAI-98 : Workshop on Learning for Text Categorization*.

Schapire, R. E. 2002. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*.

Schleimer, S.; Wilkerson, D.; and Aiken, A. 2003. Winnowing: local algorithms for document fingerprinting. In *Proceedings of SIGMOD*.

Spam, M. 2004. http://spamconference.org.

SpamAssassin. http://www.spamassassin.org/.

`spambaseftp`. ftp://ftp.ics.uci.edu/pub/machine-learning-databases/spambase/.

Vipul. http://razor.sourceforge.net.

Witten, H. I., and Frank, E. 2000. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco.