Combining Email Models for False Positive Reduction

Shlomo Hershkop
Columbia University
500 W 120 St
New York, NY 10027
shlomo@cs.columbia.edu

Salvatore J. Stolfo
Columbia University
500 W 120 St
New York, NY 10027
sal@cs.columbia.edu

ABSTRACT

Machine learning and data mining can be effectively used to model, classify and discover interesting information for a wide variety of data including email. The Email Mining Toolkit, EMT, has been designed to provide a wide range of analyses for arbitrary email sources. Depending upon the task, one can usually achieve very high accuracy, but with some amount of false positive tradeoff. Generally false positives are prohibitively expensive in the real world. In the case of spam detection, for example, even if one email is misclassified, this may be unacceptable if it is a very important email. Much work has been done to improve specific algorithms for the task of detecting unwanted messages, but less work has been report on leveraging multiple algorithms and correlating models in this particular domain of email analysis.

EMT has been updated with new correlation functions allowing the analyst to integrate a number of EMT's user behavior models available in the core technology. We present results of combining classifier outputs for improving both accuracy and reducing false positives for the problem of spam detection. We apply these methods to a very large email data set and show results of different combination methods on these corpora. We introduce a new method to compare multiple and combined classifiers, and show how it differs from past work. The method analyzes the relative gain and maximum possible accuracy that can be achieved for certain combinations of classifiers to automatically choose the best combination.

Categories & Subject Descriptors:

H.3.3 [**Information Search and Retrieval**]: Retrieval models, Selection process. H.4.3 [**Communications Applications**]: Electronic mail. I.5.3 [**Clustering**]: Similarity measures. I.6.4 [**Model Validation and Analysis**].

General Terms:

Algorithms, Performance.

Keywords:

Data Mining, Email Mining, Spam, Multiple Classifiers, Model Combination, Aggregators, False Positive Reduction.

1. INTRODUCTION

Email is undoubtedly the internet's killer application. Email is as entrenched in modern life as telephony, and offers new opportunities for various analytical tasks. An organization or user's email communication can be leveraged for many different applications in commerce, security and in managing resources in networks. In our previous work on the Email Mining Toolkit, EMT, we describe a number of these and how one may extract a wide variety of information and models of user and organizational behavior for various purposes, including virus detection. However, email can also be abused. By way of example, in this paper, we focus on the problem of Spam detection. Spam has been the subject of much debate, as well as a considerable amount of research and technology development to detect and eliminate it.

Current estimates indicate that over sixty percent of email traffic is regarded as spam and there is little reason to expect this continuous deluge will subside. Recent anti-spam laws and proposed email protocol changes aiming to restrict how and who can send emails has only had the effect of changing the way Spammers send their messages but have not decreased the amount of Spam.

In much of the research on Spam conducted to date, the research community has focused on defining a universal definition of Spam, and general-purpose methods to detect such emails. The fact remains that although 99% of user's may agree that certain emails are indeed Spam, the remaining 1% may actually wish to receive such emails. Put another way, the true positives of a spam filter for one user may actually be false positives for a different user, and vice a versa.

In our work, we define spam as emails unwanted by the user, not necessarily email deemed by a third party as unwanted. The dual is also true. Email received by some users may not be regarded by a third party as Spam, yet those emails may still be unwanted by the users in question. Our research is thus focused on learning which email a user may actually want to receive and which they do not, rather than depending entirely upon a third party arbiter to decide that central question on behalf of all users.

EMT was designed to analyze email corpora, including the entire set of email sent and received by an individual user, and to model the user's email behavior in order to classify that email for a variety of tasks, including in this case Spam detection. For example, EMT can be used to compute models under the guidance of a user to detect and classifying email into any categorization they desire. Hence, ordinary users may automatically organize and manage their own email archive, a criminal investigator may study a large corpus of email evidence more efficiently, and a network administrator may more readily determine the entry point of a new virus attack and the email clients poised for infection.

Recently, supervised machine learned methods have been heavily studied and reported in the literature to improve the accuracy of Spam filters. The prior generation of rule-based Spam filters have failed for many reasons not least of which is the error prone methods of string matching algorithms [7, 18, 31]. A great deal of the published work follows the same paradigm of training a Spam classifier given examples of Spam emails. For example, Naïve Bayes models to detect Spam was first introduced by [36] with a Bernoulli word vector model. SVM-based spam detection models [21, 24] and boosting spam models [5] were compared in [10] and genetic algorithms versus Naïve Bayes were introduced and evaluated in [20]. Text distance models [39, 40] and pattern extraction versus Naïve Bayes were compared in [33]. Further work has been done using cost-sensitive based learning [1, 2, 17], extensions to Naïve Bayes bag-of-words modeling [30, 34] and a comparative evaluations of different Naïve Bayes models [38]. An excellent overview of these methods and techniques is provided by [13, 28]. Recently even DNA pattern analysis [35] have also made it to the spam classification domain. To be sure, machine learning applied to Spam detection has received intense study over the last few years.

In this work, we evaluate methods that combine a variety of different models to determine whether we can more accurately identify what is truly Spam to a particular user, with the goal of substantially reducing the number of false positives over any individual classifier. Combining multiple classifiers to increase performance is not new. However, methods that combine and correlate multiple classifiers have not been adequately explored in the Spam detection domain. In addition, we propose a new model combination method that correlates a considerably diverse set of user email behavior models; i.e. not only email content models are used to advantage, but also user behavior models as well which do not rely upon the content of emails.

The rest of the paper is organized as follows. We first describe related work in combining multiple models in the specific case of spam detection. We then describe EMT and the individual machine learning models embedded in EMT, and methods for combining these classifiers. We then present a metric that estimates the performance of a classifier. This metric guides the means of combining multiple classifiers to produce an improved correlated system of classifiers with better overall performance than any of the component classifiers. We then detail the data sets and the performance results testing these ideas, comparing individual classifiers to the performance achieved under different classifier combinations. The paper concludes with a discussion of several open problems and future work.

2. Related Work

A considerable amount of literature exists concerning various methods for combining multiple classifiers. Combining and correlating models has been used in speech recognition, statistical pattern recognition, fraud detection, document classification, handwriting analysis and other fields. Various approaches combine models using different feature sets, other works correlate model outputs. An overview of the topic appears in [6, 22, 23]. Numerous studies have shown that combining classifiers yields better results than achievable with an individual classifier [9, 25]. Some propose combining very strong classifiers (i.e. with low error rates) [32, 43] assuming that weak classifiers (high false positives) will not combine as well, or will require too many rounds of training to achieve low error rates. Measuring the "competence" of each classifier before combining them is a common approach as in [3].

In [37] a combination of spam classifiers is proposed. That work was limited to training a committee of classifiers on a subset of labeled data, and then training a 'president' classifier using the labeled data plus the outputs from the sub-classifiers.

The method we report combines the output of individual classifiers each of which outputs a confidence score associated with the output class label. The individual classifiers are computed by distinct machine learning algorithms some of which are trained on independent features extracted from email. We detail the collection of supervised machine learning algorithm built into EMT in the following sections.

Some of the earlier work assumes a combination of classifiers with binary output (good/bad) and [9] points out that only when the classifiers have uncorrelated errors can we improve their overall assessment. We show later why the combination of confidence factors is able to achieve better results than a combination of binary classifiers.

3. An Overview of EMT

The Email Mining Toolkit developed at Columbia University has been reported elsewhere [15, 16, 41, 42]. EMT contains behavior modeling features revealing much information about individual users as well as the behavior of groups of users in an organization, and the behavior of file attachments in an email archive. A number of machine learning and anomaly detection algorithms are embedded in the system, with a convenient interface allowing a user to select a variety of features extractable from email. We summarize some of these features.

Stationary User Profiles - Histograms are used to model the behavior of a user's email account over time. Histograms are compared to find similar behavior or abnormal behavior within the same account (between a long-term profile histogram, and a recent, short-term histogram), and between different accounts.

Attachment Statistics - EMT runs an analysis on each attachment in the database to calculate a number of metrics. These include birth rate, lifespan, incident rate, prevalence, threat, spread, and death rate. They are explained fully in [4] and are used to detect unusual attachment communication indicating security breaches or virus propagations.

Similar Users - User accounts that may behave similarly may be identified by computing the pair-wise distances of their histograms (e.g., a set of Spam accounts may be inferred given a known or suspect Spam account as a model). Intuitively, most users will have a pattern of use over time, which spamming accounts will likely not follow. (Spam bots don't sleep or eat and hence may operate at times that are highly unusual.)

Group Communication (Cliques) – Communication graph-based analyses identify clusters or groups of related email accounts that frequently communicate with each other. This information is used to identify unusual email behavior that violates typical group behavior. For example, intuitively it is doubtful that a user will send the same email message across all of his social groups. A virus attacking his address book would surely not know the social relationships and the typical communication pattern of the victim, and hence would violate the user's group behavior profile if it propagated itself in violation of the user's "social cliques".

Clique violations may also indicate internal email security policy violations. For example, members of the legal department of a company might be expected to exchange many Word attachments containing patent applications. It would be highly unusual if members of the marketing department, and HR services would likewise receive these attachments. EMT can infer the composition of related groups by analyzing normal email flows and computing cliques, and use the learned cliques to alert when emails violate clique behavior.

Recipient Frequency - Another type of modeling considers the changing conditions of an email account over sequences of email transmissions. Most email accounts follow certain trends, which can be modeled by some underlying distribution. As an example of what this means, many people will typically email a few addresses very frequently, while emailing many others infrequently. Day to day interaction with a limited number of peers usually results in some predefined groups of emails being sent. Other contacts communicated to on less than a daily basis have a more infrequent email exchange behavior. These patterns can be learned through the analysis of a user's email archive over a bulk set of sequential emails.

Every user of an email system develops a unique pattern of email emission to a specific list of recipients, each having their own frequency. Modeling every user's idiosyncrasies enables the EMT system to detect malicious or anomalous activity in the account. This is similar to what happens in credit card fraud detection, where current behavior violates some past behavior patterns.

VIP Users - The recipient frequency analysis identifies the relative importance of various email users. By extending the analysis to compute the "response rates" to a user's typical recipients, one can learn the relative rank ordering of various people. Those, to whom a user responds immediately to, are likely important people in the organization.

Besides this rich collection of email user behavior, which do not rely upon any content-based analyses, EMT also provides the means of statistically modeling the content of email flows. Several of these are described in the next section.

4. Supervised Machine Learning Algorithms

For this study we apply EMT's machine learning models in the toolkit to study various means of correlating models. Specifically we used a Naïve Bayes classifier algorithm applied to "non-content" features, an N-Gram cosine model applied to the body of emails, a text classifier also applied to the body that is an adaptation of Naïve Bayes word Tokens, a standard TF-IDF model common in Information Retrieval application, a specialized "Limited N-Gram" analysis, and a specialized "URL link" model. These individual classifiers are used as a basis for the combination classifiers evaluated in our studies. Each is a machine learning algorithm used in supervised training to emit a class label and a confidence score. An EMT user has the means of specifying arbitrary class labels, and choosing from a rich set of available features that are extracted from an email archive.

We next provide an overview of each of the component supervised machine learning algorithms built into EMT and used in this study.

4.1 Non-Content Naïve Bayes Classifier (NBAYES)

Traditional machine learning modeling of email has been based on the textual content of email messages. Typically tokens are extracted from the email body and sometimes header data then processed by some machine learning algorithm.

In our prior work, we have proposed and demonstrated how non-content features can be used to profile and separate virus and Spam from normal emails [15]. The non-content features are specific static features extracted from the email envelope which are not part of the actual message body.

One of the most studied machine learning algorithms for the task of spam detection has been applying Bayesian classification to modeling the content of email.

Bayes classifiers are based on early works by [11] in the field of pattern recognition. Given an unlabeled example, the classifier will calculate the most likely classification with some degree of probability. Bayes theorem is a way of calculating the posterior probability based on prior probability knowledge.

A Naïve Bayes classifier computes the likelihood that an email is one or another class label given a set of features extracted from the training emails. The classifier is known as *naive* because it makes a naive assumption that the tokens a statistically independent. In other words, the probability of observing the combination of a specific set of features is simply the product of the probabilities. Although this is an over simplification, it greatly reduces the computational costs of estimating the conditional probabilities and in practice as been found to work as well as neural networks and decision trees.

In this classifier the set of features extracted is a set of static features including sender and recipient email names, domain names, and zones (domain ending such as com, edu, etc). In addition, the size of the message, number of recipients, number of attachments, and the MIME-type of the email are used. For continuous value features we use a multiple Gaussian estimate to estimate a probability value as in [19]. More detailed information about our classifier can be found in [15].

4.2 N-Gram Classifier

When analyzing text, one alternative to using words as tokens is to take subsequences of the data and use these subsequences as tokens. The advantage is that we do not need to define what the notion of a word is for us to analyze the text. This is ideal for example where some foreign languages which use characters instead of words.

An N-gram represents the sequence of N adjacent characters or tokens that appear in a document. We pass an N-character (or N-word) wide window through the entire email body, one character (or word) at a time, and count the number of occurrences of each distinct N-gram. For example for a 5-gram, the sequence ``Hello world" would be mapped to tokens: ``Hello", ``ello ``, ``llo w", ``lo wo", etc.

For email modeling, the algorithm works as follows. We count the number of occurrences of each n-gram for each email; this may be viewed as a document vector. Given a set of training emails, we use the arithmetic average of the document vectors as the centroid for that set. For an unknown test email, we compute the cosine distance [8] against the centroid created for the training set. If the cosine distance is 1, then the two documents are deemed identical. The smaller the value of the cosine distance, the more different the two documents are. Cosine distance is defined as:

$$D(x,y) = \sum_{j=1}^{J} x_j y_j / (\sum_{j=1}^{J} x_j^2 \sum_{k=1}^{J} y_k^2)^{1/2} = \cos\theta_{xy}$$

Here J is the total number of possible N-grams appearing in the training set and the test email. x is the document vector for a test email, and y is the centroid for the training set. x_j represents the frequency of the j^{th} n-gram (the N-grams can be sorted uniquely) occurring in the test email. Similarly y_k represents the frequency of the kth N-gram of the centroid.

4.3 Text Classifier

This classifier is a Bayes classifier based on simple word or token frequency as described in [29]. We calculate the probability of each token as seen during training using Bayes formula and assign a confidence score of the predicted class.

$$class, score = \max \begin{cases} P(Spam) \prod P(word_i \mid Spam) \\ P(NotSpam) \prod P(word_i \mid NotSpam) \end{cases}$$

4.4 Content-based Naïve Bayes (PGRAM)

Recent work by Graham [12] on the task of spam detection has floated the idea of a partial Naive Bayes approach, biased towards low false positive rates. It also uses word tokens, but filters out predefined common tokens. We incorporate this classifier in order to compare content based analysis (only over content) to our other models.

4.5 TF-IDF Classifier

This algorithm is the standard term frequency (TF) document frequency (IDF) model commonly used in Information Retrieval applications. The words that appear more frequently in a user's email likely will be key to accurate classification of emails rather than words which appear infrequently. This feature of EMT is based upon the work reported in [39, 40].

4.6 Limited N-Gram

This classifier is influenced by the work reported in [27] on finding matching files in a large file system. We adapted that work to limit the N-grams to a subset of the possible N-grams for dramatic reduction in computational expense. We calculate an integer hash value for each subsequence, and only store those which mod to some primary number. This has the effect of ignoring over 60% of the subsequences. 10-grams are modeled in the experiments reported below.

4.7 URL Modeler

Another type of behavior within email is the behavior of a typical link within an email message. The URL model allows us to profile a typical URL link found in a user's email. By modeling the typical URL link we can differentiate between wanted and unwanted email links. The algorithm was developed to compute distances between groups of universal resource locators (URLs) found in sets of emails. For the spam detection domain, non-spam messages will typically contain embedded URLs that are likely to be similar to each other and different than those occurring in Spam messages. We define a distance metric between 2 URLs x, y given the following formula:

$$Dist = 200 * \frac{\left(\sum URL_x[i] \neq URL_y[i]\right) + \left(URL_x length - URL_y length\right)/2}{URL_x length}$$

where URL_x is the longer URL. The distance is returned as a number between 0-200 where the smaller the number the closer the two URLs are to each other. We have also defined 12 types of URL's, for example URLs can be found as image links, or ftp links, or more common http links. These different types of URL links are also taken into account in the final score.

For each individual email, we group the URLs into a single cluster. While training, the clusters are formed in the following manner. All URLs are extracted from a new training example to form a single cluster. We then evaluate this cluster against all our current clusters to see if the average difference is under some threshold, if it is, we merge the two clusters. If it is not, we create a new cluster with the current set.

During testing, we simply extract all the URLs as a cluster, and find the minimum distance to any cluster in any class. This is similar to a K-nearest neighbor algorithm. This distance is then converted into a confidence score and outputted as a predicted score. For more details please refer to [14]. The end result is a measure of how unusual or familiar a URL may be in an email message given the user's prior history of emails with embedded URL's.

5. Combining Classifiers

The goal of Model combination is to leverage multiple learned experts over a given task to improve individual model performance. We may be interested in reducing errors, improving accuracy, or a combination of the two. In addition, by including the input of many types of classifiers we can protect ourselves from risk of any one classifier being compromised.

Many different studies have shown that combination classifiers either over raw features or over classifier outputs are better than any single individual classifier in the group. We now present an overview of some combination algorithms and specifically illustrate them with examples in the Spam detection domain.

In this study, to detect Spam email, we define a two-class problem, "normal" and "spam". The outputs of each of the classifiers is a class label and a confidence value in the range [0…100]. However, these outputs are coerced to a single value in the range [0…201] by a simple mapping technique to simplify the computation of correlated classifier outputs. We subtract from 100 all scores output by a classifier with label "normal" (resulting in a 0-100 range) and add 101 to any output score labeled "spam" (producing the range 101-201). Hence, the confidence interval is measured on a scale from 0 to 201 with 0 regarded as "highly normal" and 201 representing high likelihood a message is "spam". Thus, we may treat the two-value classifier output (class label and confidence) as a single number simplifying how we compute a correlation function. We refer to these outputs as *raw scores*, which are combined by a number of correlation functions.

The training regime requires some explanation. A set of emails are first marked and labeled by the user indicating whether they are Spam, or normal. This information can also be gleaned by observing user behavior (whether they delete a message prior to opening it, or move it to a "garbage/spam" folder). For our purposes here, user's provided their email files with those considered Spam placed in a special folder. Those were labeled as Spam, while all others were labeled as normal.

Once the component classifiers are applied to this labeled email corpora, the set of model outputs (the classifier raw scores mapped to the range [0…201]) are combined by a correlation function. Some of these correlation functions require a training phase. The component classifiers are tested against their training data and these model outputs are used to train the correlation function. Several were implemented and tested. We define each correlation function as follows.

5.1 Simple Averaging (Equal Weights)

As a baseline combination scheme, scores are simply averaged. This correlation function requires no training data, as it computes a final combined model output from the raw scores produced by each component classifier.

5.2 Learned Weights - Weighted Majority

The weighted majority function is an adaptation from [26]. Each of the individual classifiers is initially assigned an equal weight vote.

During training, a threshold is chosen for binary classification (correct or not) and a tally of scores is computed with the majority vote as the predicted classification. If the majority of the classifiers are correct no weights are updated. If it is incorrect, the algorithm deducts a cost β from each of the classifiers which contributed to the incorrect vote. Modeled after the work in [26] we added a term α to each weight of the correctly voting classifiers. Unlike the original algorithm, we reward classifiers which had a correct vote when the overall majority were incorrect.

During testing of an email, if the majority of weights are more confident that the example is Spam, we return the *maximum available raw score* produced by one of the component classifiers. Conversely, if the weights are more confident that the example is normal, we return the *minimum available score*.

5.3 Naïve Bayes Combination

In the Naive Bayes Combination algorithm we attempt to estimate the likelihood of an individual classifier being correct for a given score.

We can estimate this by studying a classifier's performance over a training sample. Since ground truth is known, we can measure the error rate of the classifier and its likelihood of being correct. This probability is estimated by mapping the scores computed for the training data of the classifier to pre-defined bins over the range of the raw scores. We use bins to allow us to cluster scores to achieve a high statistical sampling and reduce the amount of computation.

The number of bins, n, is a parameter. For each bin (score range), we count the number of true spam and number of true normal samples, while keeping a total count of each class label seen in the training set. Then, we estimate:

$$P(S|C_1C_2..C_n) = \frac{P(SC_1C_2..C_n)}{P(C_1C_2..C_n)} = \delta P(S)P(C_1|S)..P(C_n|S)$$

where $P(C_i|S, BIN_j) = \dfrac{\#S_j + 1}{TOTAL\_SPAM_j + NUMBERBINS}$

for the particular bin (note the smoothing terms in the formula). C_i is the ith classifier we are combining. The $P(C_i|NotSpam)$ is calculated in a similar manner. The final score is returned in the range [0-201] by normalizing the estimated likelihood, P(S), that the sample S is Spam. The normalization is computed as follows:

$$Score(S) = 201 * \frac{P(S)}{P(S) + P(NotSpam)}$$

5.4 N Dimensional Naïve Bayes Sampling

The N dimension Bayes algorithm attempts to closely model the probabilities of the combination of returned scores. We do not assume statistical Independence among the classifiers, and thus sample the training examples using an $n \times n$ matrix. Intuitively, we expect that if one of the classifiers return a high probability of Spam, and one doesn't, we can correct this particular combination by seeing what the real label was during training and learning the probabilities. In addition because we do not have classifier independence we require a small number of bins or much more data to train upon.

For example if we set the bins to size 50 the range [0…201] will map to 5 bins (0-49,50-99,100-149,150-199,200+). If we have 2 classifiers to combine, we compute a single 5X5 matrix. If we see a score of 40 from the first classifier, and 30 from the second, this will map to location (0,0). The probability can be calculated during testing by simply extracting values from the matrix seen during training in the following manner:

$$Score(S) = 201 * \frac{1 + S_{ij}}{S_{ij} + NotSpam_{ij} + NUMBERBINS}$$

where S_{ij} is the number of Spam observed and recorded in the matrix location (i,j). We use a Laplace smoothing factor of $\frac{1}{NUMBERBINS}$, where NUMBERINS is the total number of bins (n^2) we have chosen.

6. Measuring Gains from Model Correlation

There are several ways to measure the performance of the classifier combination. Zheng et al [43] proposes a novel way based on the Data Envelope Analysis (DEA) method. This analysis produces a measurement of how accurate each classifier is in correctly classifying examples. This is different than ROC convex hull measurements proposed by Provost and Fawcett in [32].

Both methods make some strong assumptions about the performance of the underlying classifiers. For example they concentrate on combining the best classifiers, trying to measure what best means. We show how even weak classifiers can be combined in our context of computing a model correlation function.

We also propose a new way to measure the gain in the context of this two class problem, i.e. Spam classifiers. If we were to calculate the maximum gain from any classifier we could measure a relative gain in comparison with this maximum. We call this empirically measured maximum possible gain the *Judge Combination*.

6.1 Judge Combination

We would like to estimate the theoretical maximum classification score available through combining raw scores. We start with a simple question, how accurate is a combination of classifiers if we only combine the minimum or maximum available raw scores from the component classifiers in the correlation function (essentially ignoring lower scores otherwise)?

Surprisingly, the answer is very accurate! We call the following algorithm the Judge Combination. Given that we know the correct classification of an email used to train the classifiers, we return the maximum available score if it is Spam and minimum if it is not Spam.

The accuracy achieved by this combination method, is used as the theoretical limit of a possible combination algorithm and we scale all our gain results based on this accuracy estimate.

6.2 Gain formula

We define the gain as a convenient measurement of how much of its accuracy potential a classifier has reached. This measurement can be used to decide which combination algorithm to use in a system.

$$classifier\_gain = \sum_{i=0}^{10,000} \frac{(10,000 - i)}{10,000} FP_i$$

where FP_i is the measured false positive rate of the classifier over its training data. i is varied by 10,000 to allow two decimal precision for measuring the false positive rates from 0 to 100. This is simply the area under the ROC curve, biased towards lower false positive values. We can calculate the FP_i by moving a threshold over the data and calculating a ROC curve, and then averaging the results between points to interpolate the graph.

Since the Judge algorithm represents the maximum possible combination score achievable, we use it to scale the gain score for each individual classifier as follows:

$$gain = \frac{classifier\_gain}{Judge\_gain}$$

7. Experimental Setup

A large data set of real email was used to study the model combination methods. The data set consists of emails collected from five users at Columbia University spanning from 1997 to the present, a user with a hotmail account, and a user with a Verizon.net email account. In total we collected 320,000 emails. Users indicated which emails where Spam by moving them to specific folders. These are emails unwanted by each user.

Because current Spam levels on the internet are estimated at 60%, we sampled the set of emails so that we have a 60% ratio of Spam to normal. We were left with a corpus of 278,274 emails time ordered as received by each user.

We tested the models using the familiar 80/20 rule, 80% being the ratio of training to testing. Hence, the first 80% of the ordered email are used to train the component classifiers and the correlation functions, while the following 20% serve as the test data used to plot our results. This set up mimics how such an automatic classification system would be used in practice. As time marches on, emails received are training data used to upgrade classifiers applied to new incoming data. Those new data would be used as training for another round of learning to update the classifiers.

Table 1 Individual Classifier Performance

Classifiers	Detection	False Positive	Gain
Ngram	75%	4%	72.2%
URL	55%	10%	32%
TextClassifier	91%	5%	71%
Pgram	87%	2%	77.2%
Limited Ngram	66%	5%	61.4%
Nbayes(non content)	88%	3.8%	79.8%
TF-IDF	74%	4.2%	61.5%

Ideally, since some of the correlation functions require training data, we would like to train the combination algorithm concurrently with the training of the individual classifiers. Although we could have first given each classifier all the training examples and then extracted scores over those examples, we felt that would not reflect the real world setting where only partial examples of Spam would ordinarily be available, only those seen to date by the user. To this end, we batched the training data into 1000 examples used to train the classifiers. After each batch, each classifier was executed to generate raw scores for each of the examples seen in the current batch. These scores were then input to the model combination algorithms to train the correlation function. We used a batch size of 1000 for efficiency purposes, although any size should be acceptable to achieve comparable results. We realize, however, that individual classifiers will shift individual scores depending upon the amount of training data used. For example, for some classifier, its scoring might be different if we train 5000 emails rather than say 1000. However, we preferred to mimic a more realistic training regimen reflecting how such a system may actually be used as a real application, and thus we believe this batch approach is not unreasonable.

The data used was pristine and unaltered. No preprocessing was done to the bodies of the emails with the exception that all text was stropped to lower case. Headers of the emails were ignored except for subject lines that are used in some of the non-content based classifiers. While adding header data would have improved individual classification, there is much variability in what is seen in the header, and we felt it might over train and learn some subtle features of tokens only available in the header data present in the

Columbia data set. For the Ngram, TF-IDF, PGram, and Text Classifier, we truncated the email parts so that we only used the first 800 bytes of each part of the email attachment. This was used for both efficiency and computational considerations. In addition the increase in detection was about 10% over using full email bodies. The reason is because of noise in the number of tokens seen.

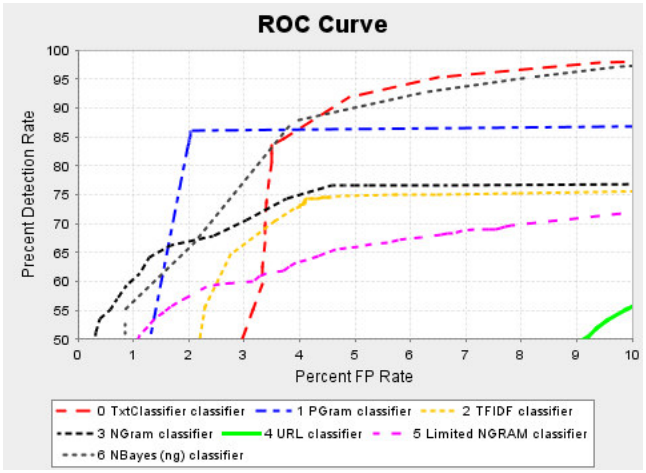

Figure 1 - Results of Individual Classifiers

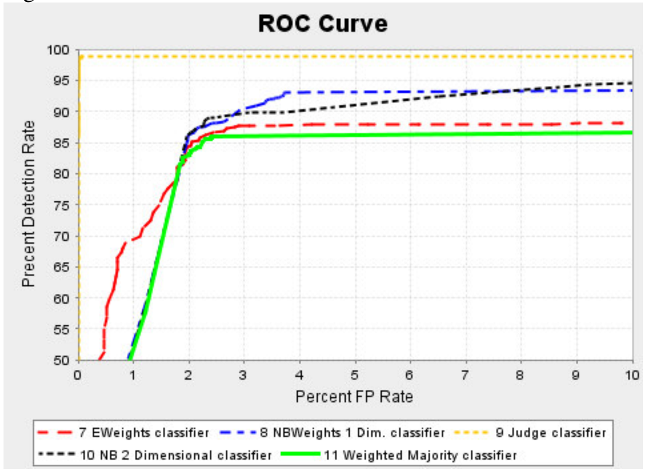

Figure 2 - Results of Combination Algorithms

8. Results

Figure 1 shows the performance results of the individual classifiers over the email data set. Of particular interest is that the NBayes non-content, Text Classifier, and PGram classifiers are all very strong classifiers. Table 1 has the detection rates highlighted at certain points to give a sense of how well they compare to each other.

Figure 2 shows the results of combining the classifiers. Table 2 has the combination algorithms and the gains achieved in each case. Notice that false positives have been reduced by about 3% and detection improved by about 4% over the best individual classifiers. This is also reflected in about a 15% improvement in the gain measurement.

Table 2 - Highlight from combination algorithms

Classifiers	Detection	False Positive	Gain
Equal Weights	87%	2.3%	84%
Single Dimension NB	93%	3.6%	85.1%
Judge Combination	99%	0.025%	-
N Dimension	88.7%	2.3%	84.5%
Weighted Majority	85.5%	2.5%	79.9%

We next compare the merit of only combining strong or weak classifiers. In Figure 5 we combined the three strongest algorithms, namely non-content, text classifier, and pgram. Notice about a 2% false positive reduction is achieved over the strongest component classifier.

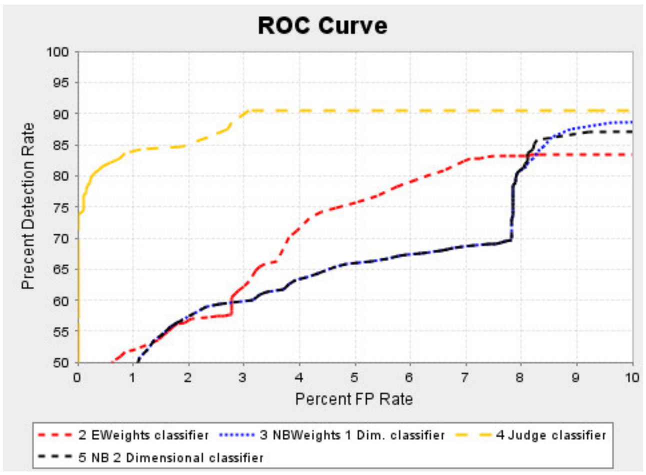

Figure 3 - Combining Ngram and Ngram-Limited

We compare the combination of Ngram, URL, Limited Ngram in Figure 6. Surprisingly the weighted majority and NB1 are almost the same here. Since TF-IDF on a full email body has a very low detection rate, we tried a combination of URL, TF-IDF(full), and Limited Ngram in Figure 7. Although there is a negligible improvement in false positive rate, there is a very strong detection improvement of about 10%.

In Figure 3 we highlight what happens when combining two similar classifiers. Notice that the performance of the Judge algorithm has been significantly reduced, as expected. This confirms that the individual classification errors strongly overlap, thus the maximum combination is also lower.

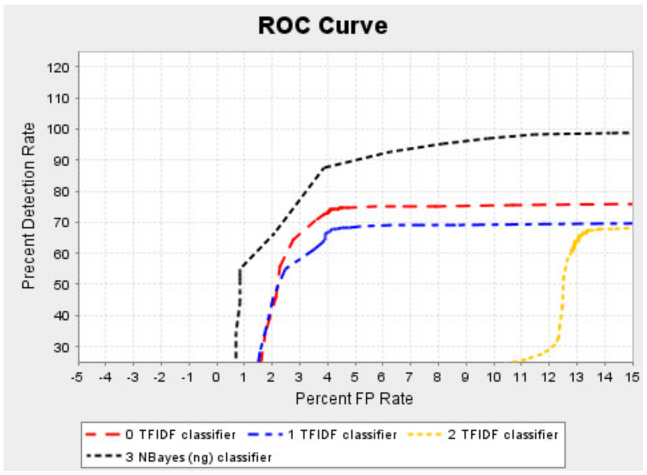

Figure 4 - TFIDF performance

The TF-IDF(full body) classifier was a surprisingly poor performer and thus represents a weak classifier combination. We show case what happens to the TF-IDF algorithm as it is exposed to less of the email parts. The improvements are going from full, to first 800, to first 400 bytes and compared to NBayes in figure 4. We also show the combination when we combine the best and worst classifiers in Figure 8. Notice that the Judge isn't returning the theoretical limit and the reason is that TF-IDF doesn't return a confidence score, but rather a distance metric to some document centroid. In addition because it is being trained on the entire body, the algorithm is being overwhelmed by noisy token probabilities.

Because of this, the scores here are not easily combined by a simple combining scheme. On the other hand, the Naïve Bayes combinations are mapping the scores to a probability space, where there can be interpreted as a confidence value. This is also the reason that it has been found that combining confidence scores, works better than combining binary classification. There is inherently more information in such cases.

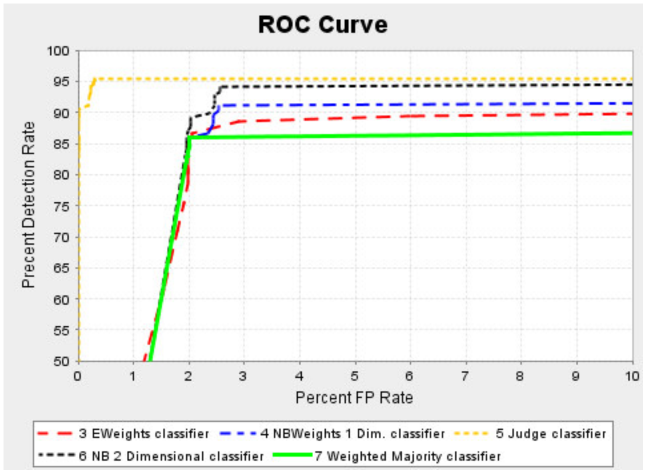

Figure 5 - Combining 3 Strong Algorithms

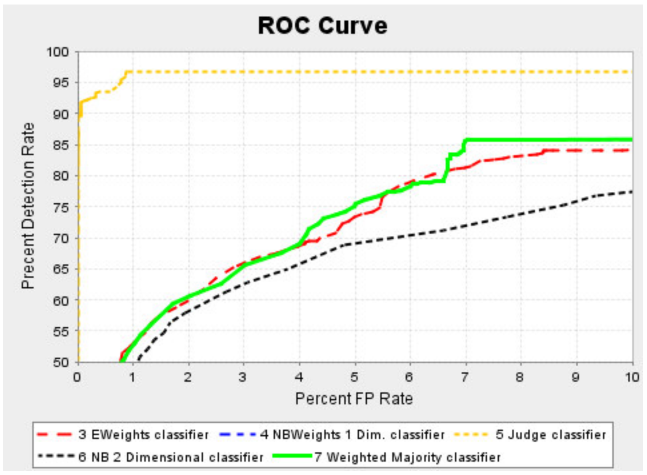

Figure 6 - Combining Weak Classifiers

9. Conclusion

We have investigated several particular model combination algorithms and plotted their performance using a number of supervised machine learning classifiers. We inspected several of the emails that were persistently misclassified by all the combinations. These peculiar emails comprised messages with only graphics attached, flaming debates on some news groups, and some empty emails that had no bodies (their message was essentially their subject line). We believe that much lower false positive rates can be achieved on this data set by including features extracted from the email header data.

In summary we have shown how combination algorithms can be directly applied to Spam classifiers, and used to improve both false positive and detection rates on either strong, weak, or a combination of these classifiers. We rely on the gain formula to help choose a specific algorithm.

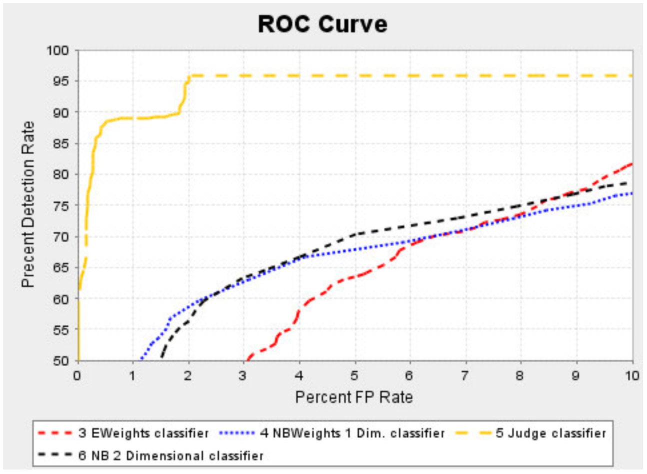

Figure 7 - Combining Weak Classifiers

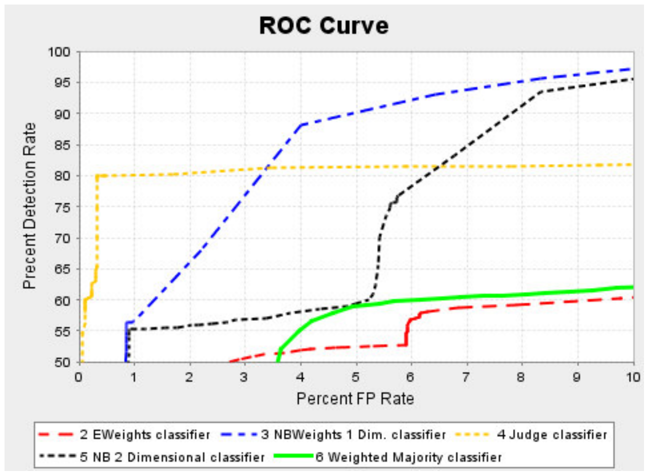

Figure 8 – Combining Non content Naïve Bayes and TF-IDF(full body)

A very strong reason to work with combination algorithm is something learned from the computer security domain: resistance to attack. Concentrating on only one single classifier, to fine tune it to perfection, will only encourage spammers to try to, and eventually defeat, that specific mechanism. By using a combination algorithm, an email enclave can be protected against compromise of any single Spam classifier. This is highlighted in Figure 8 where the scores of the TF-IDF (over full body) can be thought of being compromised in some fashion so that they do not reflect a true confidence. The combination algorithms which are based on remapping probabilities are remarkably resilient in face of this kind of attack. In reality, a simple equal weights algorithm can be used, and a second probability algorithm run along side as a reality check on the performance of the classifier. Having automatic checks and balances, is one way in which spam classification can gain user confidence by displaying a confidence metrics, without requiring the user to trudge through mounds of spam in the spam folder searching for misclassified examples.

In future work, we would like to combine these classifiers with some of the open source Spam filters and perhaps to automatically update the Spam filters when EMT has developed sufficient evidence for a flood of new Spam. Even so, EMT is being developed as an integrated solution extending an ordinary email client program so that users can *train a filter* to make their own judgments about email Spam.

The results achieved indicate that EMT is a fairly robust and accurate system providing a powerful tool for a variety of analysis tasks. These concepts are applicable to a far wider range of problems, including virus detection, security policy violations, and a host of other detection tasks. It is important to note that testing EMT in a laboratory environment only suggests what its performance may be on specific tasks and source material. The behavior models are naturally specific to a site or particular account(s) and thus performance will vary depending upon the quality of data available for modeling, and the parameter settings and thresholds employed. Although the model combination functions work well in these test cases, there is still other analytical tasks that should be explored to determine how to fully automate a model combination and correlation function feature. In the case of the two-class Spam detection problem, the methodology is straightforward. This may not be the case for multi-class learning problems.

Table 3 - Combining Strong Classifiers

Classifiers	Gain
Text Classifier	73%
PGram	80.1%
Naïve Bayes Non content	82.8%
Equal weight	84.6%
NB 1 dimension	84%
NB n Dimension	86.6%
Weighted Majority	80.2%

Table 4 - Weak Classifiers

Classifier	Gain
Ngram	74.3%
URL	32.0%
Limited Ngram	63.25%
Equal Weights	73%
Naïve Bayes 1	74%
NB n Dimension	65%

10. ACKNOWLEDGMENTS

Special thanks to the IBM Anti-Spam team for letting us work with them on developing these ideas. Specifically we would like to thank J. Kephart, R. Segal, M. Wegman, V. T. Rajan, and J. Ossher. This research was partially supported by an NSF grant *Email Mining Toolkit Supporting Law Enforcement Forensic Analyses* from the Digital Government research program, No. 0429323.

11. REFERENCES

1. Androutsopoulos, I., Koutsias, J., Chandrinos, K., Paliouras, G. and Spyropoulos, C. An Evauation of Naïve Bayesian Anti-Spam Filtering.
2. Androutsopoulos, I., Koutsias, J., Chandrinos, K. and Spyropoulos, C., An experimental comparison of naive bayesian and keywordbased anit-spam filtering with personal email messages. in *23rd annual international ACM SIGIR conference on Research and development in information retrieval*, (2000), 160-167.
3. Asker, L. and Maclin, R., Ensembles as a Sequence of Classifiers. in *15th International Joint Conference on Artificial Intelligence*, (Nagoya, Japan, 1997), 860-865.
4. Bhattacharyya, M., Hershkop, S., Eskin, E. and Stolfo, S.J., MET: An Experimental System for Malicious Email Tracking. in *New Security Paradigms Workshop (NSPW-2002)*, (Virginia Beach, VA, 2002).
5. Carreras, X. and Mrquez, L., Boosting trees for anti-spam email filtering. in *RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing*, (Tzigov Chark, BG, 2001).

6. Clemen, R.T. Combining forecasts: A revew and annotated bibliography. *International Journal of Forecasting*, *5*. 559 - 583.
7. Cohen, W., Learning rules that classify e-mail. in *Machine Learning in Information Access: AAAI Spring Symposium (SS-96-05)*, (1996), 18-25.
8. Damashek, M. Gauging Similarity via N-Grams: Language-Independant Sorting, Categorization and Retrieval of Text. *Science*, *267*. 843-848.
9. Dietterich, T.G. Ensemble Methods in Machine Learning. *Lecture Notes in Computer Science*, *1857*. 1-15.
10. Drucker, H., Wu, D. and Vapnik, V.N. Support Vector Machines for Spam Categorization. *IEEE Transactions on Neural networks*, *10* (5).
11. Duda, R. and Hart, P. *Pattern classification and scene analysis*. John Wiley & Sons, New York, 1973.
12. Graham, P. A Plan For Spam, 2003.
13. Hallam-Baker, P. A Plan For No Spam, Verisign, 2003.
14. Hershkop, S. Using URL Clustering to Classify Spam, Columbia University, 2005.
15. Hershkop, S. and Stolfo, S.J. Identifying Spam without Peeking at the Contents. *ACM Crossroads*.
16. Hershkop, S., Wang, K., Lee, W. and Nimeskern, O. Email Mining Toolkit Technical Manual, Computer Science Dept, Columbia University, New York, 2004.
17. Hidalgo, J.M.G. and Sanz, E.P., Combining Text and Heuristics for Cost-Sensitive Spam Filtering. in *Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*, (Lisbon, 2000).
18. Itskevitch, J. Automatic Hierarchical E-Mail Classification Using Association Rules, 2001.
19. John, G. and Langley, P., Estimating continuous distributions in Bayesian classifiers. in *Eleventh Conference on Uncertainty in Artificial Intelligence*, (1995), 338-345.
20. Katirai, H. Filtering Junk E-Mail: A Performance Comparison between Genetic Programming and Naive Bayes, 1999.
21. Kiritchenko, S. and Matwin, S., Email Classification with Co-Training. in *CASCON 2001*, (2001).
22. Kittler, J. and Alkoot, F.M. Sum versus Vote Fusion in Multiple Classifier Systems. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, *25* (1).
23. Kittler, J., Hatef, M., Duin, R.P.W. and Matas, J. On Combining Classifiers. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, *20* (3).
24. Kolcz, A. and Alspector, J., SVM-based Filtering of E-mail Spam with Content-specific Misclassification Costs. in *Workshop on Text Mining (TextDM'2001)*, (San Jose, California, 2001).
25. Larkey, L.S. and Croft, W.B., Combining Classifiers in Text Categorization. in *SIGIR-96: 19th ACM International Conference on Research and Development in Information Retrieval*, (Zurich, 1996), ACM Press, NY, US, 289-297.
26. Littlestone, N. and Warmuth, M.K. The Weighted Majority Algorithm. *IEEE Symposium on Foundations of Computer Science*.
27. Manber, U., Finding Similar Files in a Large File System. in *Usenix Winter*, (San Fransisco, CA, 1994), 1-10.
28. Massey, B., Thomure, M., Budrevich, R. and Long, S., Learning Spam: Simple Techniques for Freely-Available Software. in *USENIX 2003*, (2003).
29. Mitchel, T. *Machine Learning*. McGraw-Hill, 1997.
30. Peng, F. and Schuurmans, D., Combining Naive Bayes and n-Gram Language Models for Text Classi cation. in *25th European Conference on Information Retrieval Research (ECIR)*, (2003).
31. Pollock, S. A rule-based message filtering system. *ACM Trans. Office Automation Systems*, *6* (3). 232-254.
32. Provost, F. and Fawcett, T. Robust Classification for Imprecise Environments. *Machine Learning*, *42*. 203-231.
33. Provost, J. Naïve-Bayes vs. Rule-Learning in Classification of Email, 1999.
34. Rennie, J., ifile: An Application of Machine Learning to E-mail Filtering. in *KDD-2000 Workshop on Text Mining*, (2000).
35. Rigoutsos, I. and Huynh, T., Chung-Kwei: a Pattern-discovery-based System for the Automatic Identification of Unsolicited E-mail Messages. in *ceas 2004*, (Mountain View, California, 2004).
36. Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E., A Bayesian approach to filtering junk e-mail. in *AAAI-98 Workshop on Learning for Text Categorization*, (1998).
37. Sakkis, G., Androutsopolous, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. and Stamatopoulos, P., Stacking classifiers for Anti-Spam Filtering of Emails. in *6th conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, (2001).
38. Schneider, K.M., A Comparison of Event Models for Naive Bayes Anti-Spam E-Mail Filtering. in *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, (Budapest, Hungary, 2003).
39. Segal, R.B. and Kephart, J.O., Incremental Learning in SwiftFile. in *17th International Conf. on Machine Learning*, (San Francisco, CA, 2000), Morgan Kaufmann, 863--870.
40. Segal, R.B. and Kephart, J.O., MailCat: An Intelligent Assistant for Organizing E-Mail. in *3rd International Conference on Autonomous Agents*, (1999).
41. Stolfo, S.J., Hershkop, S., Wang, K., Nimeskern, O. and Hu, C.-W. A Behavior-based Approach to Securing Email Systems. *Mathematical Methods, Models and Architectures for Computer Networks Security*.
42. Stolfo, S.J., Hershkop, S., Wang, K., Nimeskern, O. and Hu, C.-W., Behavior Profiling of Email. in *1st NSF/NIJ Symposium on Intelligence & Security Informatics(ISI 2003)*, (Tucson, Arizona, 2003).
43. Zheng, Z., Padmanabhan, B. and Zheng, H., A DEA Approach for Model Combination. in *KDD2004*, (Seattle, WA, 2004).