Data Mining Methods in the Detection of Spam

Dong-Her Shih
Hsiu-Sen Chiang
Chia-Shyang Lin
National Yunlin University of Science and Technology

Ming-Hung Shih[1]
National Chiao Tung University

The spam problem has generated enormous costs for companies and users of the Internet. Internet users not only pay for the bandwidth to bring in volumes of spam mail but also pay for its storage. In this paper, we propose a modified Naïve Bayesian classifier and compare it with three data mining methods for identifying whether incoming mail is spam or legitimate automatically. The experimental results show that although there is no dominant algorithm to the spam problem, generally the decision tree has the better performance. Our proposed modified Naïve Bayesian classifier has the potential for further investigation as well.

Introduction

Unsolicited Bulk Email (UBE), also referred to as Unsolicited Commercial Email (UCE), is commonly called spam or "junk mail." Spamming is the practice of sending mass mailings to large numbers of people who have no relationship with the sender and who didn't request the mail. According to research conducted by Microsoft and published by the Radicati Group, the percentage of spam mail in the total number of emails sent daily has been consistently growing since 2005. As a result, spam is expected to represent 77% of emails sent worldwide by the end of 2009, amounting to almost 250 billion unsolicited emails delivered every day.

[1]The authors would like to give thanks to the National Science Council of Taiwan for their grant (NSC93-2213-E-224-038) to perform part of this research.

Competitive anti-spam products, legislation, and efforts towards a better user education have all been used in an attempt to stop spam. However, unsolicited emails keep consuming the space and time of all email users. Moreover, spam messages can be the cause of serious virus and spyware outbreaks, while others "phish" for sensitive information like bank accounts and passwords. SPAMHAUS says that spammers are carrying out a dictionary attack on hotmail.com. The spammers connect to victims' mail servers and submit millions of random email accounts in common words and names (e.g. michaelFxy2@_.com, marla1892@_.com), recording which addresses succeed and add these automatically to their list. They also send spam to the variation of the account name (e.g. Lidia@_.com to 1idia@_.com) without collecting the victims' email accounts (Spammers Grab MSN Hotmail addresses, 2007).

As email becomes an important medium of communication, with direct impact on human relations and business, spam email causes considerable damage to the users and the entire internet foundation. For instance, organizations such as those in the financial and healthcare industries are required by law to archive email for up to seven years. They not only pay for the bandwidth to bring in volumes of spam mail but also pay for its storage (Paulson, 2003).

Due to the worsening spam problem, several studies have been conducted, ranging from technical to legal. In the technical respect, filtering is currently the most widely used method. The filter can be implemented on either the server end (mail transport agent, MTA) or the user's end (mail user agent, MUA). Motivated by those previous works, in this study, we propose an enhanced Naïve Bayesian classifier method and compare it with three data mining methods in order to identify whether incoming mail is spam or legitimate automatically: ID3 Decision Tree, Naive Bayesian Filter, and R.A Fisher's Probability Combination Method. The performance-measuring result shows that the ID3 decision tree has better performance, in general.

In the next section, we will briefly describe some related methods used to fight spam. Then, we will outline the three approaches: ID3 Decision Tree, Naive Bayesian, and R.A Fisher's Probability combination method, and our enhanced Naive Bayesian methods in section 3. The experimental method and the performance measures will be provided in section 4. We will show and analyze the experiment results in the last section of this paper, and also give a brief conclusion.

Related Methods

During these years, the CAN-SPAM Act of 2003 became the most well known out of all the regulatory solutions. The CAN-SPAM Act seeks to control rather than outright ban spam by filtering and forbidding deceptive email messages, which include either misleading email headers (routing information), often referred to as "header forging," fake return addresses, or misleading subject lines.

Legal scholars also note that CAN-SPAM's greatest deficiency is that it supersedes and nullifies much stricter state laws. The CAN-SPAM Act may be an important step for the spam regulation, but must be amended in order to provide further protection. In a world where spam holds such an important position, methods of preventing it should also be given increasing importance. To combat spam, multiple filtering technologies

have been developed that weed out most, but not all of the unsolicited email.

Blacklist: A blacklist spam filter can be a Domain Name System-based (DNS-based) or email-address-based blacklist. A DNS-based blacklist is a means by which an Internet site may publish a list of IP addresses that some people may want to avoid, in a format that can be easily queried by computer programs on the Internet. The technology is built on top of the Internet DNS. DNSBLs are chiefly used to publish lists of addresses linked to spamming. Most mail transport agent (mail server) software can be configured to reject or flag messages which have been sent from a site listed on one or more such lists. The email address-based blacklist utilizes the full email address of the user (i.e. a domain-based identifier plus the local part). Since the local part can also be spoofed, validation mechanisms must be in place. Examples of those are Lightweight Directory Access Protocol (LDAP) and Active Directory (Alperovitch, Judge & Krasser, 2007).

Blacklist is very useful at the ISP level, but it has several weaknesses also. First, more than half of the spam mail servers are not on the blacklist. Second, the effect of the blacklist depends on the administrator of the blacklist. If the blacklist is wrong, it is possible that legitimate emails may also get filtered in the process.

Signature-based Filtering: The method of signature-based filtering compares incoming email with the spam that has already been received. In order to know whether two emails are the same, the filter calculates "signatures" for them. Signature-based filters rarely block legitimate mails, but its weakness is that spammers can add elements to each email and give it a distinct signature, thereby tricking the signature-based filters.

Rule-Based Filtering: Rule-based filters try to discover the patterns found in many spam messages (e.g. words or phrases, malformed headers and misleading dates). SpamAssassin, a popularly used open-source spam filter, uses a large set of heuristic rules. But the main disadvantage of rule-based filters like SpamAssassin is that they tend to have high false-positive rates (O'Brien & Carl, 2003).

Text Classification Filtering: A text classification filter uses the text classification technique to filter spam. There have been several studies done on this application, including keyword-based, phrase-based, and character-based studies. The Naïve Bayes-based method is also another efficient approach of keyword and phrase-based studies which use features extracted from emails. Additionally, Support Vector Machine (SVM), Rocchio, and decision tree filtering based on the ID3, C4.5, or C5 algorithms can be identified as the representative methods to analyze keywords in email (Schapire & Singer, 2000; Drucker, Wu & Vapnik, 1999; Joachims, 1997; Quinlan, 1993).

More recently, text categorization techniques are being applied in anti-spam research. As mentioned earlier, the Naïve Bayesian classifier is a widely used method. This paper attempts to demonstrate the performance of the Naïve Bayesian classifier method by using the concept of integrated multi-attribute and also by incorporating information Gain (IG) techniques in extracting and computing the weights of feature terms.

Methodology

In this section we will describe three different data mining methods that are used to generate classifiers that detect whether incoming mail is spam or legitimate. These algorithms include Naïve Bayes, R. A. Fisher's probability combination method, and ID3 decision tree. The Naive Bayesian method has been used several times before to filter spam email (Androutsopoulos et al., 2000; Mehran et al., 1998) and has overall been very effective. On the other hand, when a Bayes-like method is proposed, it can release the independent assumption by combining Graham and Fisher's method of filtering spam (Robinson, 2003). We took Robinson's approach for generating probabilities associated with words, altered it slightly, and proposed a Bayesian calculation for dealing with words that hadn't appeared very often in the spam messages. Then, we took the approach based on the chi-square distribution for combining the individual word probabilities and turned it into a combined probability representing an email. In order to distinguish spam from useful email efficiently, we adopted an ID3 decision tree to produce some rules. Using those rules is an easy way to ensure that only valid emails reach recipients and can educate users to help in preventing spam distribution.

Naïve Bayes Classifier

A Naïve Bayes classifier computes the likelihood that an email is spam given the features that are contained in the email itself. Assuming that there were similar contents in spam emails that differentiated them from legitimate emails, the class of legitimate emails had similar patterns that differentiated them from the spam emails. The model output by the Naïve Bayes algorithm labels emails based on their contents.

The Naïve Bayes algorithm computes the probability that a given feature is spam and the probability that a feature is legitimate by computing statistics on the set of training data. Then, to predict whether a mail is spam or legitimate, those probabilities are computed in the classifier and the Naïve Bayes independence assumption is used. The independence assumption is then applied in order to efficiently compute the probability that an email was spam or legitimate.

In the Naïve Bayes anti-spam method, each mail is represented by a vector $\vec{x}$=< x_1, x_2, x_3, ...,x_n >, where x_1, x_2, x_3, ...,x_n are the values of attributes X_1, X_2, X_3, ...,X_n. As shown previously, we following Sahami, et al. (1998) and using binary attributes, i.e. X_i = 1 if the email has the property represented by X_t (in our case, a specific word), and X_i = 0 otherwise.

Given the vector $\vec{x}$=< x_1, x_2, x_3, ...,x_n >,of email, and where $k \in$ {*spam, legitismate*}, the probability that an email belongs to category c is:

$$P(C = c \mid \overrightarrow{X} = \vec{x}) = \frac{P(C = c) \cdot P(\overrightarrow{X} = \vec{x} \mid C = c)}{\sum_{k \in \{spam, legitimate\}} P(C = k) \cdot P(\overrightarrow{X} = \vec{x} \mid C = k)} \quad (1)$$

Androutsopoulos, et al. (2000) notes that the probabilities $P(\vec{X} \mid C)$ are almost impossible to calculate, because the possible values of vector X are too many and there

are also data sparseness problems. The Naïve Bayes filter assumes that X_1, X_2, X_3, ...,X_n are conditionally independent given the category C, which allows us to calculate $P(C = c \mid \vec{X} = \vec{x})$ as:

$$P(C = c \mid \vec{X} = \vec{x}) = \frac{P(C = c) \cdot \prod_{i=1}^{n} P(X_i = x_i \mid C = c)}{\sum_{k \in \{spam, legitimate\}} P(C = k) \cdot \prod_{i=1}^{n} P(X_i = x_i \mid C = k)} \quad (2)$$

$P(X_t \mid C)$ and $P(C)$ are easy to estimate from the frequencies of the training data. A large number of empirical studies have found the Naïve Bayes filter to be surprisingly effective, despite the fact that the assumption that X_1, X_2, X_3, ...,X_n are conditionally independent is usually overly simplistic (Domingos & Pazzani, 1996; Langley, Wayne & Thompson, 1992).

Fisher's Probability Combination Method

Robinson (2003) proposed a Bayes-like method that can release the independent assumption through R. A. Fisher's method to combine probability. For each word that appears in the training data we calculate:

$$b(w) = \frac{\text{number of spam containing the word } w}{\text{total number of spam}} \quad (3)$$

$$g(w) = \frac{\text{number of legistimate mail containing the word } w}{\text{total number of legistimate mail}} \quad (4)$$

$$p(w) = \frac{b(w)}{b(w) + g(w)} \quad (5)$$

$p(w)$ can be interpreted as the probability that a randomly chosen email address containing the word "w" will be spam. There is a problem with the probabilities calculated above as when some words are very rare in the training set. For instance, if a word appears in exactly one email and is a spam, the value of $p(w)$ is 1.0. Clearly, it is not a good idea to classify all future emails that contain that same word as spam. In fact, the situation is such that we simply don't have enough data to know the real probability.

Virtually any word can appear in either a spam or non-spam message, and those of data points are not enough to be completely certain that we know the real probability. The Fisher's probability combination approach lets us combine our general background information with the data we have collected for a word in such a way that both aspects are given their proper importance. In this way, we determine an appropriate degree of belief about whether, when we see the word again, it will be in a spam message. We calculate this degree of belief, $f(w)$, as follows:

$$f(w) = \frac{(s \times x) + (n \times p(w))}{s + n} \quad (6)$$

where:

s: the strength we want to give to our background information
x: our assumed probability, based on our general background information, that a word we don't have any other experience of will first appear in a spam
n: the number of emails we have received that contain word

In practice, the values for *s* and *x* are found through testing to optimize performance. Reasonable starting points are 1 for s and 0.5 for *x*.

In the proposed method, we should first calculate $(-2)1n(p_1 \text{ x } p_2 \text{ x...x } p_n)$. Then, consider the result to have a Chi-square with 2n degrees of freedom, and use Chi-square table to compute the probability. The "spamness" probability of an email that contains specific w is:

$$H = C^{-1}[-2\ln\prod_{w} f(w), 2n] \tag{7}$$

where:

H: the "spamness" probability of a mail
C^{-1} : the inverse Chi-square function, used to derive a p-value from a Chi-square distributed random variable.

ID3 Decision Tree

A decision tree is similar to a flow chart. In order to classify an unknown sample, the attribute values of the sample are tested against the decision tree. A path is traced from the root to a leaf node that holds the class prediction for that sample. Decision trees can easily be converted to classification rules. Decision trees have been used in many application areas ranging from medicine, to game theory and business. The basic algorithm used in decision tree induction is the greedy algorithm which constructs decision trees in a top-down, recursive, and divide-conquer manner. The algorithm summarized in Figure 1 is ID3, a well-known decision tree induction algorithm (Han & Kamber, 2001).

The information gain measure is used to select the test attribute at each node in the tree. Such a measure is referred to as an attribute selection measure or a measure of the goodness of split. The attribute with the highest information gain (or greatest entropy reduction) is chosen as the test attribute for the current node. This attribute minimizes the information needed to classify the samples in the resulting partitions and reflects the least randomness or "impurity" in these partitions. Such an information-theoretic approach minimizes the expected number of tests needed to classify an object and guarantees that a simple (but not necessarily the simplest) tree is found.

Figure 1: *Basic algorithm for inducing a decision tree from training samples*

Algorithm: Generate_decision_tree. Generate a decision tree from the given training data.

Input: The training samples, represented by discrete-valued attributes, the set of candidate attributes, attribute-list.

Output: A decision tree.

Method:

```
create a node N;
if samples are all of the same class, C then
  return N as a leaf node labeled with the class C;
if attribute-list is empty then
  return N as a leaf node labeled with the most common class in samples; // majority voting
select test-attribute, the attribute among attribute-list with the highest information gain;
label node N with test-attribute;
for each known value a_i; of test-attribute // partition the samples
  grow a branch from node N for the condition test-attribute = a_i;
  let s_i be the set of samples in samples for which test-attribute = a_i; // a partition
  if s_i is empty then
  attach a leaf labeled with the most common class in samples;
else attach the node returned by Generate_ decision_ tree (s_i, attribute-list-test-attribute);
```

Let S be a set consisting of s data samples. Suppose the class label attribute has m distinct values defining m distinct classes, C_i (for i=1,…,m). Let s_i be the number of samples of S in class C_i. The expected information needed to classify a given sample is given by:

$$I(s_1,s_2,...s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{8}$$

Where p_i is the probability that an arbitrary sample belongs to class C_i and is estimated by s_i /s. Note that a log function to the base 2 is used since the information is encoded in bits. Let attribute A have v distinct values, $\{a_1, a_2, \ldots, a_v\}$. Attribute A can be used to partition S into v subsets, $\{S_1, S_2, \ldots, S_v\}$, where S_j contains those samples in S that have value a_j of A. If A were selected as the test attribute (i.e., the best attribute for splitting), then these subsets would correspond to the branches grown from the node containing the set S. Let a_{ij} be the number of samples of class C_i in a subset S_j. The entropy, or expected information based on the partitioning into subsets by A, is given by

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j},\ldots,s_{mj}) \tag{9}$$

The term $\dfrac{S_{1j} + \ldots + S_{mj}}{S}$ acts as the weight of the *jth* subset and is the number of

samples in the subset (i.e., having value a_j of A) divided by the total number of samples in S. The smaller the entropy value is, the greater the purity of the subset partition. Note that for a given subset S_j,

$$I(s_{1j}, s_{2j}, \ldots s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2(p_{ij}) \tag{10}$$

Where $p_{ij} = \frac{S_{ij}}{|S_j|}$ and is the probability that a sample in S_j belongs to class C_i.

The encoding information that would be gained by branching on A is

$$Gain(A) = I(s_1, s_2, \ldots s_m) - E(A) \tag{11}$$

In other words, *Gain(A)* is the expected reduction in entropy caused by knowing the value of attribute A. The algorithm computes the information gain of each attribute. The attribute with the highest information gain is chosen as the test attribute for the given set S. A node is created and labeled with the attribute, branches are created for each value of the attribute, and the samples are partitioned accordingly. When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers. Tree pruning methods address this problem of over fitting the data. Such methods typically use statistical measures to remove the least reliable branches, generally resulting in faster classification and an improvement in the ability of the tree to correctly classify independent test data.

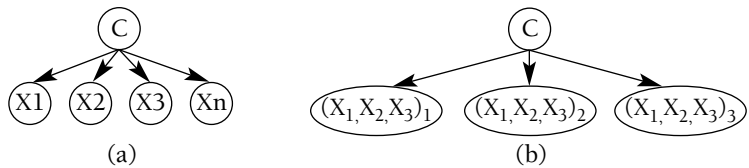
A Novel Modified Naïve Bayesian Classifier

As we mentioned previously, the probabilities $P(\vec{X} \mid C)$ in equation (1) are almost impossible to calculate, because the possible values of vector *X* are too many and we may not have enough data to derive the value. Under the independent assumption, the Naïve Bayesian filter allows us to compute $P(\vec{X} \mid C)$ as $\prod P(X = x \mid C = c)$. For example, we change our view to the vector $\vec{x}$=< x_1, x_2, x_3,..., x_n > into $\vec{x}$'=< (x_1, x_2, $x_{3)1}$,...,(x_1, x_2, $x_{3)k}$ >, then we calculate $P(C = c \mid \vec{X} = x)$ as:

$$P(C = c \mid \overrightarrow{X'} = \vec{x}) = \frac{P(C = c) \cdot \prod_{j=1}^{k} P[(x_1, x_2, x_3)_j \mid C = c]}{\sum_{k \in \{spam,\, legitimate\}} P(C = k) \cdot \prod_{j=1}^{k} P[(x_1, x_2, x_3)_j \mid C = c]} \tag{12}$$

The main difference is the posterior probability needed by the classifier. Figure 2 contrasts the structure of the Naïve Bayesian Classifier with our modified Bayesian Classifier.

Figure 2: *Naïve Bayesian model (a), and the modified Naïve Bayesian model (b)*



The logic behind our method is that the more dimensions of the feature space, the more accuracy of conditional probability we can get. It will also give us the chance to derive more information from the email content.

Experiments

In this section, we will use the Spam Email Database from the UCI Machine Learning Repository to train and test the algorithms previously described.

Data Set

The Spam Email Database was created by Hewlett-Packard Labs (UCI, 2004). It had been used for the HP internal-only technical report and other spam detection studies. The database contains 4601 instances. Each instance has 58 attributes (57 continuous, 1 nominal class label for the identification of spam and legitimate mail). In order to test the algorithms, we choose the 48 word attributes and transform the continuous attributes into 1 (has the specific word of the attribute) or 0 (has not). We randomly choose 50% instances (2282 instances) for the algorithm training and 2319 instances for the testing. The data set and its usage in our study is summarized as follows:

Table 1: *Descriptive statistics and the usage of the data set*

Total	
4601 instances Spam: 1813 (39.4%) Legitimate: 2788 (60.6%)	
Training Data	Testing Data
2282 instances Spam: 905 (39.7%) Legitimate: 1377 (60.34%)	2319 instances Spam: 908 (39.15%) Legitimate: 1411 (60.85%)

Naïve Bayes

In order to calculate the "spamness" through Naïve Bayes method, first, we have to calculate the posteriori probability with the training data, as shown in Table 2.

Table 2: *The posteriori probability (For Naïve Bayes)*

X_i	make	address	free	business	credit	money	...	order
$P(X_i = 0 \mid SPAM)$	64.5%	66.4%	45.0%	62.3%	78.8%	61.5%	...	58.4%
$P(X_i = 1 \mid SPAM)$	35.5%	33.6%	55.0%	37.7%	21.2%	38.5%	...	41.6%
$P(X_i = 0 \mid Legitimate)$	86.0%	89.8%	91.1%	90.5%	98.2%	88.4%	...	98.8%
$P(X_i = 1 \mid Legitimate)$	14.0%	10.2%	8.9%	9.5%	1.8%	11.6%	...	1.2%

After the posterior was derived from the training data, we can calculate the probability of an email being spam through its feature vector and equation (2). The performance of the Naïve Bayes filter will be compared with other filters below.

Fisher's Method

In Robinson (2003), the posterior probability needed for the R.A Fisher's method is different from the Naïve Bayes. It has two more benefits than Naïve Bayes: (1) it can deal with the rare word problem we mentioned in 3.2; (2) the probability combination needs no attribute independent assumption, and is more reasonable for the reality. As shown in Table 3, we calculate the posteriori probability through equation (3), (4), and (5). And the degree of belief, *f(w)*, is derived from the equation (6).

Table 3: *The posteriori probability p(w) and degree of belief f(w)*

w_i	make	address	free	business	credit	money	...	order
$p(w_i)$	71.7%	76.7%	86.1%	79.8%	92.1%	95.8%	...	81.3%
$f(w_i)$	71.7%	76.6%	86.0%	79.8%	91.9%	95.6%	...	81.2%

The combination of probability for a specific feature vector of email is derived through equation (7) with the degree of belief, *f(w)*.

Decision Tree

Instead of using a complicated Bayesian calculation to extract a simple rule set from our data, we have introduced the decision tree induction method for the classification of spam and legitimate emails. Given the generalized and relevant data relations, the information gain for each candidate attribute can be computed using the algorithm in Figure 1. The candidate attribute that gives the maximum information gain as the decision attribute at this current level is selected and the current set of objects are partitioned accordingly. For each subset created by the partitioning, it is necessary to repeat each step to further classify data until either (a) all or a substantial proportion (no less than the classification threshold) of the objects are in one class, (b) no more attributes can be used for further classification, or (c) the percentage of objects in the subclass (with respect to the total number of training samples) is below the exception threshold. The decision tree for spam email detection has been developed. The knowledge represented in the decision tree can be extracted and represented in the form of IF-THEN rules. One rule is created for each path from the root to a leaf node. Figure 3 shows the sample of the rules.

Figure 3: *The IF-THEN rule of the decision tree output*

IF *"remove"="YES", "money"="YES", "hp"="YES", "project" ="NO"* **THEN**
LEGITIMATE
IF *"remove"="NO", "money"="YES", "hp"="NO", "business"="YES"* **THEN**
SPAM
IF *"remove"="NO", money="YES", "hp"="NO", "business"="NO", "edu"="YES"* **THEN**
LEGITIMATE
IF *"remove"="NO",money="YES",hp="NO",business="YES",edu="NO",george="YES"* **THEN**
SPAM

Comparisons and Analysis

Some very promising results were returned from the three algorithms, as can be seen in Table 4. To evaluate our system we were interested in several quantities typically used in measuring the query result of information retrieval. These are: (1) True Positives (TP): the number of spam email classified as spam, (2) True Negatives (TN): the number of legitimate email classified as legitimate, (3) False Positives (FP): the number of legitimate emails falsely classified as spam and (4) False Negatives (FN): the number of spam emails falsely classified as legitimate.

The performance of the algorithms can be measured in terms of accuracy rate. The precision rate, TP / (TP +FP), denotes the portion of spam in the filtered mail, while the recall rate, TP / (TP +FN), answers the question what portions of spam can the algorithm filter. The accuracy rate, (TP +TN) / (TP +FP + FN + TN), represents the overall correct decision of the filtering.

Table 4: *Results of the experiments*

	True Positives	True Negatives	False Positives	False Negatives	Precision	Recall	Accuracy
Naive Bayes	31%	56%	2%	11%	94%	74%	87%
Fisher's Method	36%	53%	5%	6%	89%	87%	90%
Decision Tree	38%	55%	3%	4%	92%	91%	93%

The Naïve Bayes has the highest precision rate, but the recall and accuracy rates are not as good as others and often suffer from the false negatives rate. The Fisher's method has better recall and accuracy rates than Naïve Bayes, though the precision rate is the lowest of the three. The decision tree method generally has better performance than the others.

Proposed Novel Modified Naïve Bayesian Classifier

To introduce cost-sensitive evaluation to our method, we employ the weighted accuracy (*WAcc*) which was proposed by Androutsopoulos (2000). We treat each legitimate email as if it is λ emails. When a legitimate email is wrongly blocked, we will count it as λ errors. When a legitimate email passes the filter, we will count it as λ successes. These lead to the definition of the *WAcc* and weighted error rate (*WErr*). We assume λ = 9,99. The result of our experiment is presented in Table 5. As can be seen from Table 6, the proposed method has better performance than Naïve Bayesian Filter.

$$WAcc = \frac{\lambda \cdot Tn + Tp}{\lambda \cdot (Tn + Fp) + Tp + Fn} \tag{13}$$

Table 5: *Results of the extra experiments*

Experiment	Method	TP	FP	TN	FN	Detection	False Positive	Overall Accuracy	Precision
a	Naïve Bayesian	348	55	642	79	81.5%	7.9%	88.08%	86.35%
	Modified Naïve Bayesian	349	43	655	77	81.92%	6.16%	89.32%	89.03%
b	Naïve Bayesian	364	53	678	77	82.54%	7.25%	88.91%	87.29%
	Modified Naïve Bayesian	364	44	687	77	82.54%	6.02%	89.68%	89.22%

Table 6: *Weighted results of the extra experiments*

$\lambda = 9$	Experiment	Methodology	WAcc	WErr = 1 - WAcc
9	a	Naïve Bayesian	87.06%	12.94%
		Modified Naïve Bayesian	88.52%	11.48%
	b	Naïve Bayesian	87.68%	12.32%
		Modified Naïve Bayesian	88.7%	11.3%
99	a	Naïve Bayesian	91.62%	8.38%
		Modified Naïve Bayesian	93.33%	6.67%
	b	Naïve Bayesian	92.26%	7.74%
		Modified Naïve Bayesian	93.47%	7.53%

Conclusion

The spam problem has generated enormous costs for companies and other users of the Internet, and it continues to worsen. In order to deal with the huge amount of spam people receive daily, powerful email filters with high reliability are needed. In this study, we examined various ways to stop spam. Three data mining methods in the detection of spam were described and examined. From the results, we have found that it is possible to train the filter automatically through data mining algorithms. In our experiment results, the decision tree method generally had better performance than the other two methods. The main contribution of this study was to provide a clear performance measure of three data mining methods and propose a novel modified Naïve Bayesian classifier in advance. Although spam email detection and protection is to be viewed as necessary for all Internet users, there is no single method that can dominate this work. We also explored several methods for spam filtering, ranging from social to technical approaches. One of the most important areas of future work is the development of more efficient algorithms. The current data mining methods require a significant amount of memory and computing resource. We would like to make these learning algorithms more efficient overall.

References

Alperovitch, D., Judge, P. & Krasser, S. (2007). Taxonomy of Email Reputation systems. *27th International Conference on Distributed Computing Systems Workshops:* 27.

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V. & Spyropoulos, C. D. (2000). An Experimental Comparison of Naïve Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages. *In Proc. of the 23rd Annual International ACM SIGR Conference on Research and Development in Information Retrieval.* Athens, Greece: 160–167.

Cheng, J. & Greiner, R. (1999). Comparing Bayesian Network Classifiers. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (UAI-99). San Francisco, CA: 101-108.

O'Brien, C. & Vogel, C. (2003). Spam filters: Bayes vs. chi-squared; letters vs. words. *In Proceedings of the International Symposium on Information and Communication Technologies*. Dublin, Ireland: 291-296.

Drucker, H., Wu, D. & Vapnik, V. N. (1999). Support Vector Machines for Spam Categorization. *IEEE Transaction on Neural Networks, 10*(5): 1048-1054.

Domingos P. & Pazzani M. (1996). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. I*n Proc. of the 13th International Conference on Machine Learning*. Bari, Italy: 105-112.

Han, J. & Kamber, M. (2001). Data mining concepts and techniques. USA: Morgan Kaufmann: 284–287.

Joachims, T. (1997). A probabilistic analysis of the Ricchio algorithm with TFIDF for text categorization. *In Proc. of 14th International Conference on Machine Learning.*

Langley, P., Wayne, I. & Thompson, K. (1992). An Analysis of Bayesian Classifiers. *In Proc. of the 10th National Conference on Artificial Intelligence.* San Jose, California: 223-228.

Metzger, J., Schillo M. & Fischer, K. (2003). A multi-agent based peer-to-peer network in java for distributed spam filtering. *In Proc. of the CEEMAS,* Czech Republic: 616-625.

Paulson, L. D. (2003). Group Considers Drastic Measures to Stop Spam. *IEEE Computer, 36*(7): 21- 22.

Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. *Morgan Kaufmann series in machine learning.*

Robinson, G. (2003). A Statistical Approach to the Spam Problem. *Linux Journal, 107*: 3.

Sahami, M., Dumais, S., Heckerman, D. & Horvitz, E. (1998). A Bayesian Approach to Filtering Junk Email. *In Learning for Text Categorization – Paper from the AAAI Workshop,* Madison Wisconsin. AAAI Technical Report WS-98-05: 55-62.

Schapire, R.E. & Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning, 39:* 135–168.

Spammers Grab MSN Hotmail addresses. 2007 [Online]. Available: *http://www.spamhaus.org/news.lasso?article=6.*

UCI Machine Learning Repository. 2004 [Online]. Available: *http://www.ics.uci.edu/~mlearn/MLRepository.html.*