Chapter 1 INTRODUCTION

1.1 Problem of Spam

Electronic mail, often abbreviated as e-mail or email, is any method of creating, transmitting, or storing primarily text-based human communications with digital communications systems.

Spam is the abuse of electronic messaging systems to indiscriminately send unsolicited bulk messages. Email spam, know as unsolicited bulk Email (UBE) or unsolicited commercial email (UCE), is the practice of sending unwanted e-mail messages, frequently with commercial content, in large quantities to an indiscriminate set of recipients.

Spam in e-mail started to become a problem when the Internet was opened up to the general public in the mid-1990s, it grew exponentially over the following years, and today comprises some *80%* to *85%* of all the email in the world, by conservative estimate; some sources go as high as *95%*. *(Kanich, et al., 2008)*

As the number of users connected to the Internet, electronic mail (E-mail) is quickly becoming one of the fastest and most economical forms of communication available. Since e-mail is extremely cheap and easy to send, it has gained enormous popularity not simply as a means for letting friends and colleagues exchange messages, but also as a medium for conducting electronic commerce. Unfortunately, the same virtues that have also attracted direct marketers to bombard unsuspecting e-mail boxes with unsolicited messages regarding everything from items for sale and get-rich-quick schemes to information about accessing pornographic web sites.

With the proliferation of direct marketers on the Internet and the increased availability of enormous e-mail address mailing lists, the volume of spam mail has grown tremendously in the past few years. As a result, many readers of e-mail must now spend a lot of time on-line wading through such unwanted messages. Moreover, since some of these messages can contain offensive material (such as graphic pornography), there is

often a higher cost to users of actually viewing this mail than simply the time to sort out the spam. Lastly, spam mail not only wastes user time, but can also quickly fill-up file server storage space.

There is an immediate need to control the progressively growing spam flood. A great deal of on-going research is trying to resolve the problem. However, e-mail users are impatient and therefore there is a growing need for rapidly available anti-spam solutions to protect them.

1.2 Why Naive Bayesian ?

Naive Bayesian spam filter is a well-known tool for spam filtering which belongs to Statistical Classifier. The Naive Bayesian classifier has shown good results based on its accuracy than other well-known classifiers such as – Neural Network (NN), Support Vector Machines (SVM) and J48. *(Youn, Seongwook and McLeod, Dennis, 2006)*

Table 1.2: Classification result based on data size

Data Size	NN	SVM	Naive Bayesian	J48
1000	93.50%	92.70%	97.20%	95.80%
2000	97.15%	95.00%	98.15%	98.25%
3000	94.17%	92.40%	97.83%	97.27%
4000	89.60%	91.93%	97.75%	97.63%

The Naive Bayesian spam filter is trained before testing and classification of any incoming emails. Both ham and spam emails are used for training the classifier. While training, every email is broken down into tokens after converting into lowercase. Then the tokens or words, which are not significant, are removed – called feature selection process. The selected tokens are stored in database for calculation of probability of each token that being a spam or ham. Thus obtained probabilities of tokens and the number of occurrences of tokens are the knowledge for the spam filter. In the testing phase, for classification of an incoming email, the email is tokenized at first. After that, the probability of each token 'being spam' is calculated based on prior knowledge. Finally, the probability of the email 'being spam' is decided on the basis of the probability of the tokens containing by that email.

The purpose of this research work is to a comparative study of the performance of Naive Bayesian spam filter using two different tokenization methods: word distribution and trigrams.

1.3 Research Objectives

The objective of this thesis is to explore the statistical filter called Naive Bayesian Spam filter based on word distribution and trigrams tokenization. After dissecting the segments of its operation, this work focuses on following specific areas described below.

- To compare the efficiency of the spam filter based on two different tokenizing methods.
- To examine how different probability estimators affect the spam classification performance.
- To conduct comparative analyses between the results of Naive Bayesian Spam Filter based on Word Distribution verses Trigrams.

Chapter 2 LITERATURE REVIEW

Various machine learning techniques have been proposed to identify and filter-out spam such as rule-learning (*Cohen, 1996*), Naive Bayes (*Sahami, et al., 1998*), Decision Trees (*Carreras & Marquez, 2001*), Support Vector Machines (*Drucker, et al., 1999*), Memory-Based Learning (*Androutsopoulos, et al., 2000a*), or a combination of these techniques which are currently used by real-world filtering software to identify and filter spam. For instance, *SpamAssasin*[1] uses a combination of IP Address blacklists, Bayesian filtering, online databases and checksum-based spam filtering. Other examples include *SpamBayes*[2] and *Bogofilter*[3] which both use a Naive Bayesian approach for filtering spam. The basic concept behind these techniques is in the classification of emails using a trained classifier that can automatically predict that a certain message is spam or legitimate. This automatic process would increase filtering performance and better usability than manual classification of emails.

In a paper published by Microsoft researchers (*Sahami, et al., 1998*), an approach made use of the Bayesian-based approach to spam filtering was discussed and evaluated. The approach made use of the Bayesian theorem by tokenizing words contained in email, classifying them based on a corpus of words into spam and ham emails as feature variables, and predicting the probability of messages being spam or ham based on the combined probability of feature variables in the received message. The approach basically applied the Bayesian network into a classification task. Additionally, *Sahami and others (1998)* used both words and phrases in the tokenization stage and testing whether the filter improved as a result of using the combination. Then they added domain specific properties and evaluated the effectiveness of the resulting spam filter. Domain specific properties – whether textual or non-textual – are added as feature vectors in a similar methodology used with words and phrases. Properties such as a message originating from a domain name ending with a ".edu" or the ratio of non-alphanumeric characters in emails are used for or against the classification of mail into spam or ham. *Sahami and others*

[1] See http://spamassassin.apache.org

[2] See http://spambayes.sourceforge.net

[3] See http://bogofilter.sourceforge.net

(1998) used the Space Vector Model (*Salton & McGill, 1983*) to represent messages as feature vectors where every dimension corresponds to a word in the entire corpus. Table 1 shows the results of *Sahami and others (1998)*.

Table 2.1: Classification results using various feature sets (Sahami, et al., 1998)

Feature Regime	**Junk**		**Legitimate**	
	Precision	**Recall**	**Precision**	**Recall**
Words only	97.1%	94.3%	87.7%	93.4%
Words + Phrases	97.6%	94.3%	87.8%	94.7%
Words + Phrases + Domain-Specific	100.0%	98.3%	96.2%	100.0%

It is hypothesized that the use of trigrams in the tokenization of email contents would increase the accuracy and reliability of the classifier due to the higher potential for discrimination associated with trigrams and as a result the efficacy of Bayesian-based spam filters would increase. The efficacy of a Bayesian-based spam filter is compared between word-based tokenization and trigram-based tokenization. The comparison will be done by implementing a Naive Bayesian classifier using two different tokenizers. The first tokenizer uses single words whilst the second tokenizer uses trigrams (three-word phrases). A suitable approach to trigram-based tokenization will be developed. The classifier would be the same in both instances. The implementations will use a public corpus of spam and legitimate emails and will test the two implementations on a set off messages from the same corpora of emails. The results will be analyzed and any findings will be discussed.

Lai and Tsai (2004) compared the performance of various machine learning methods for spam email categorization. Their results showed strong evidence that header information improved the filtering performance of all the tested classifiers whether they were based on a Naive Bayesian methodology or other methodologies such as Term Frequency-Inverse Document Frequency and Support Vector Machine. Additionally, stemming had no effect on the performance of spam filters. Stemming refers to algorithms that reduce words to their base or root and is a concept generally used in search engines and indexing. Moreover, when they tested the filtering ability based only on the subject or the body of emails, poor performance was observed. As such, the results

from the study strongly suggested the use of all information contained in the email from the header to and including the body.

A study on web search (*Johnson, et al., 2006*), investigated the effects of bigrams and trigrams. The results from the study showed good indicators that bigrams and trigrams are more suitable identifiers of text, and as such, they qualify as high quality discriminators in text categorization. *Johnson and others (2006)* referred to previous research in natural language processing in favor of the use of phrases due to the higher meaning conveyed in them as opposed to single-words. They concluded that bigram and trigram based search queries improved the efficacy of searching. Moreover, another study by *Lyon and other (2006)* examined the use of trigrams in the identification of plagiarized text. Based on the assumption that the English language follows a *Zipfian distribution (1949)*, *Lyon and others (2006)* used trigrams for tokenization. By analyzing corpora of text from TV news, Federalist papers, and the Wall Street Journal, they showed that trigrams provide a high measure of uniqueness suitable for text identification. Additionally, they noted that this observed phenomenon is not unique to European languages and that trigrams made from Chinese terms share the same level of uniqueness observed in the English language. *Lyon and others (2006)* tested the identification of plagiarized programming code using text identifiers based on trigram tokenization.

Chapter 3 TECHNIQUES TO ELIMINATE SPAM

There are several approaches which deal with spam. Some common spam filtering techniques are as follows:-

3.1 Hiding the e-mail Address

The simplest approach to avoid spam is to keep the e-mail address hidden from spammers. The email address can be revealed only to trusted parties. For communication with less trusted parties a temporary e-mail account can be used. If the e-mail address is published on a web page it can be disguised for e-mail *spiders*[4] by inserting a tag that is requested to be removed before replying. Robots will collect the e-mail address with the tag, while humans will understand that the tag has to be removed in order to retrieve the correct e-mail address. For most users this method is insufficient. Firstly, it is time consuming to implement techniques that will keep the e-mail address safe, and secondly, the disguised address could not only mislead robots, but also be unreachable from other people like relatives, acquaintances etc. Once the e-mail address is exposed, there is no further protection against spam.

3.2 Pattern Matching, Whitelists and Blacklists

This is a content-based pattern matching approach where the incoming e-mail is matched against some patterns and classified as either spam or legitimate. Many e-mail programs have this feature which is often referred to as “message rules” or “message filters”. This technique mostly consists of a plain string matching. Whitelists and blacklists, which basically are lists of friends and enemies, fall into this category. Whenever an incoming e-mail is matched against an entry in the whitelist, the rule is to allow that e-mail through. However whenever an e-mail has a match against the blacklist, it is classified as a spam. This method can reduce spam up to a certain level and requires constant updating as spam evolves. It is time consuming to determine what rules to use and it is hard to obtain good results with this technique. In *Mertz D. 2002* some simple rules are presented. The author claims that he was capable of catching about *80%* of all spam he received. However, he stated that the applied rules had shown high *false positive*

[4] E-mail spiders, or e-mail robots, are computer programs that scans and collects e-mail address from Internet.

rates. Basically, this technique is a simpler version of the more sophisticated “rule based filters”.

3.3 Rule Based Filters

This is a popular content-based method deployed by spam filtering software such as *SpamAssassin*[5]. Rule-based filters apply a set of rules to every incoming email. If there is a match, the e-mail is assigned a score that indicates spaminess or non-spaminess. If the total score exceeds a threshold the e-mail is classified as spam. The rules are generally built up by regular expressions and they come with the software. The rule set must be updated regularly as spam changes, in order for the filtering of spam to be successful. Updates are retrieved via the Internet. The tests results from the comparison of anti-spam programs presented in *Holden 2003* had shown that *SpamAssasin* found about *80%* of all spam, while statistical filters find close to *99%* of all spam. The advantage of rule-based filters is that they require no training to perform reasonably well. Rules are implemented by humans and they can be very complex. Before a newly written rule is ready for use, it requires extensive testing to make sure it only classifies spam as spam and not legitimate messages as spam. Another disadvantage of this technique is the need for frequent updates of the rules. Once the spammer finds the way to deceive the filter, the spam messages will get through all filters with the same set of rules.

3.4 Statistical Filters

In *Sahami et al. 1998*, it is shown that it is possible to achieve remarkable results by using a statistical spam classifier. Since then many statistical filters have appeared. The reason for this is simple; they are easy to implement, have a very good performance and require a little maintenance. Statistical filters require training on both spam and non-spam messages and will gradually become more efficient. They are trained personally on the legitimate and spam e-mails of the user. Hence it is very hard for a spammer to deceive the filter.

[5] SpamAssasin, http://www.spamassassin.org/index.html

3.5 E-mail Verification

E-mail verification is a *challenge–response system*[6] that automatically sends out a one-time verification e-mail to the sender. The only way for an e-mail to pass through the filter is if the sender successfully responds to the challenge. The challenge in the verification e-mail is often a hyperlink for the sender to click. When this link is clicked, all e-mails from that sender are allowed through. *Bluebottle*[7] and *ChoiceMail*[8] are two such systems. The advantage of this method is able to filter almost *100%* of the spam. However, there are two drawbacks associated with this method. The sender is required to respond to the challenge which needs extra care. If this challenge is not recognized the e-mail will be lost. Verifications can also be lost due to technical obstacles such as firewalls and other e-mail response systems. It can also cause problems for automated e-mail responses such as online orders and newsletters. The verification e-mail also generates more traffic.

3.6 Distributed Blacklists of Spam Sources

These filters use a distributed blacklist to determine whether or not an incoming e-mail is spam. The distributed blacklist resides on the Internet and is frequently being updated by the users of the filter. If a spam passes through a filter, the user reports the e-mail to the blacklist. The blacklist is updated and will now protect other users from the sender of that specific e-mail. This class of blacklists keeps a record of known spam sources, such as IP numbers that allow SMTP relaying. The problem involved in using a filter entirely relying on these blacklists is that it will generally classify many legitimate e-mails as spam (*false positive*). Another disadvantage is the time taken for the networked based lookup. These solutions may be useful for companies assuming that all their e-mail communications are with other serious non-listed businesses. Companies offering this service include *MAPS*[9], *ORDB*[10] and *Spamcop*[11].

[6] It is a type of spam filter that automatically sends a reply with a challenge to the suspected sender of an incoming e-mail.
[7] Bluebottle, http://www.bluebottle.com/
[8] ChoiceMail, http://www.digiportal.com/
[9] Mail Abuse Prevention System LLC (MAPSSM), http://mailabuse.com/
[10] Open Relay DataBase (ORDB), http://ordb.org/
[11] Spamcop, http://www.spamcop.net/

3.7 Distributed Blacklist of Spam Signatures

These blacklists work in a same manner to "distributed blacklist of spam sources". The difference is that these blacklists consist of spam message signatures instead of spam sources. When a user receives a spam, that user can report the message signature (typically a hash code of the e-mail) to the blacklist. In this way, one user will be able to warn all other users that a certain message is spam. To avoid non-spam being added to a distributed blacklist, many different users must have reported the same signature. Spammers have found an easy way to fool these filters; they simply add a random string to every spam. This will prevent the e-mail from being detected in the blacklist. However spam fighters attempt to overcome this problem by adapting their signature algorithms to allow some random noise. The advantage being that these kinds of filters rarely classify legitimate messages as spam. The greatest disadvantage is that they are not able to recall much of the spam. *Vipul's Razor* uses such a blacklist and states that it catches *60%-90%* of all incoming spam. Another disadvantage is the time taken for the network lookup.

3.8 Money e-mail Stamps

The idea of e-mail stamps is not new, but it is not until recently that major companies have considered using it to combat spam. The sender would have to pay a small fee for the stamp. This fee could be minor for legitimate e-mail senders, while it could destroy business for spammers that send millions of e-mails daily. There are two stamp types; money stamps and proof-of-work stamps. *GoodmailSystems*[12] is developing a system for money stamps. The basic idea is to insert a unique encrypted *id* to the header of each sent e-mail. If the recipient ISP is also participating in the system, the *id* is sent to *Goodmail* where it is decrypted. *Goodmail* will now be able to identify and charge the sender of the e-mail. Today there are many issues requiring solutions before such a system can be deployed. Who receives the money? Where is tax paid? Who are allowed to sell stamps?

3.9 Proof-of-work e-mail Stamps

At the beginning of 2004, Bill Gates, Microsoft's chairman, suggested that the spam problem could be solved within two years by adding a proof-of-work stamp to each

[12] Goodmail, http://www.goodmailsystems.com/

e-mail. *Camram*[13] is a system that uses proof-of-work stamps. Instead of taking a micro fee from the sender, a cheat-proof mathematical puzzle is sent. The puzzle requires a certain amount of computational power to be solved (matter of seconds). When a solution is found, it is sent back to the receiver and the e-mail is allowed to pass to the receiver. The puzzle *Camram* is using is called *Hashcash*[14].

Whether it is money or proof-of-work e-mail stamps, many oppose the idea, not only because emailing should be free, but also because it will not solve the spam problem. To make this approach effective, most ISP's would have to join the stamp program. As long as there are ISP's that are not integrated into the stamp system, spammers could use their servers for mass e-mailing. It could then still be possible for the legitimate e-mailers to pay to send e-mails, while spam is still flooding into the inboxes of users. Many non-profit legitimate mass e-mailers will probably have to abandon their newsletters due to the sending cost. Historically, spammers have been able to deceive most of the other anti spam filters and this could also be the case with the stamp system.

3.10 Legal Measures

In recent years many nations have introduced anti-spam laws, in December 2003, president George W. Bush signed the *CAN-SPAM*[15] act, the Controlling the Assault of Non-Solicited Pornography and Marketing Act. The law prohibits the use of forged header information in bulk commercial email. It also requires spam to include opt-out instructions. Violations can result in fines of $250 per e-mail, capped at $6 million. In April 2004 the first four spammers were charged under the *CANSPAM* law (*Rainie, L., & Fallows, D., 2004).* The trial is still on, but if the court manages to send out a strong message, this could deter some spammers. The European Union introduced an anti-spam law on the 31st of October 2003 called "The Directive on Privacy and Electronic Communications". This new law requires that companies gain consent before they send out commercial e-mails. Many argue that this law is toothless since most of the spam comes from the outside of EU. In the long-run legislation can be used to slowdown the spam flood to some extent, but it will require an international movement.

[13] Camram, http://www.camram.org/
[14] Hashcash, http://www.hashcash.org/
[15] http://www.spamlaws.com/

Chapter 4 STATISTICAL CLASSIFIERS

A classifier's task is to assign a pattern to its class. The pattern can be a speech signal, an image or simply a text document. For example in spam classification, the classifier would assign a message as either spam or legitimate class.

Historically, rule-based classifiers were mainly used until the end of the 1980s. Rule-based classifiers are simple but require classification rules to be written. Writing rules for high accuracy is difficult and time consuming. By the end of 1980s, when computers were becoming more efficient, statistical classifiers started to emerge. Statistical classifiers use machine learning to build its classifier from previously labeled (the class is known) training data. For example, a statistical spam classifier is trained on labeled legitimate and spam messages and a speech recognition classifier is trained on different labeled voices. The classifier uses characteristics of the pattern to classify it into one of several predefined classes. Any characteristic can be referred as a feature.

4.1 Features and Classes

A feature is any characteristic, aspect, quality or attribute of an object. For example, the eye color of a person or the words in a text documents are features. A good feature is one that is distinctive for the class of the object. For example, the word 'Viagra' is found in many spam messages but not in many legitimate, hence it is a good feature. In most cases many features makes the classification more accurate. The combination of n features can be represented as an n -dimensional vector, called a feature vector. The feature vector is defined as $F=\{f_1, f_2 ,..., f_n \}:1\leq i\leq n$ where f_i is a feature. The n -dimensionality of the feature vector is called the feature space. By examining a feature vector the classifier's task is to determine its class. If m is the number of classes, then the class vector is defined as $C=\{c_1,c_2 ,..., c_m \}$, where $c_k ,1\leq k\leq m$ is a unique class.

4.2 Text Categorization

Text categorization is the problem involved in classifying text documents to a category or class. Text categorization is becoming more popular as the amount of digital textual information grows. The problem of classifying an e-mail message as spam or

legitimate message can be considered as a text categorization problem. Another popular area of use is Web page categorization to hierarchical catalogues.

Statistical text classifiers can be divided into two categories - generative and discriminative. The generative approach uses an intermediate step to estimate parameters while the discriminative models the probability of a document belonging to a class directly. There are arguments for using discriminative methods instead of involving the intermediate step of generative approaches. Recent studies (*Ng and Jordan 2002*) have shown that the performances of generative and discriminative approaches are highly dependent on the corpus training data size.

SVM (*Vapnik 1995*) separates two classes with vectors that pass through training data points. The separation is measured as the distance between the support vectors and is called the margin. The time involved in finding support vectors that maximize the margin is, in the worst-case scenario, a quadratic. SVM have shown promising results concerning text categorization problems in several studies (*Yang Y. & Liu X, 1998*). A recent study (*Androutsopoulos 2004*) demonstrated that its performance was good with reference to the spam domain.

Another classifier, k-Nearest Neighbor (kNN), maps a document to features and measures the similarity to the k-nearest training documents. Scores are created for each of the classes of the k-nearest documents based on the similarity. The document is then classified as the class with the greatest similarity. This approach has been available for over four decades and has proved to be the top-performer on Reuters corpus (topic classification of text documents).

Neural Networks (NNet) is commonly used in pattern analysis and has been applied to text categorization by *Wiener et al. 1995* & *Yang Y. and Liu X, 1998*. In a study (*Chen D. et al.*) the NNet was outperformed by NB. NNets are expensive to train and memory consuming as the number of features grow.

Among the described classifiers the NB classifier is the simplest in terms of its ease of implementation. When compared to the others, it is also computationally efficient. Tests carried out by *Yang Y. and Liu X, 1998* showed that NB is better than others.

Another study (*Androutsopoulos 2000a*) shows that NB is better than kNN. For this work NB is used as classifier not only for its simplicity and computational efficiency, but also because of a belief that with a good probability estimator and careful feature selection.

4.3 Basics about Probability Theory

The probability that an event *X* occurs is a number that can be obtained by dividing the number of times *X* occurs by the total number of events. The probability is always between 0 and 1, or it can be expressed as percentage. For example, the probability of a six sided die showing 6 is $P(6) = \frac{1}{6}$ or it is approximately equal to16.67%. Two events are independent if they do not affect each other's probabilities. For example, the events "tossing a coin" and "rolling a die" are independent because the probability of the coin landing on its head is not affected by the probability of rolling a six on a die.

For independent events, the probability of both occurring is called a joint probability and it is calculated as a product of the individual probabilities. The probability for event *X* is *P(X)* and for event *Y* is *P(Y)* . If *X* and *Y* are independent, then their joint probability is expressed as

$$P(X \quad Y) = P(X).P(Y) \tag{1}$$

Using the example with the coin and die, the probability that the coin lands on head and the six is rolled is $P(HEAD).P(6) = \frac{1}{2} \mid \frac{1}{6} = \frac{1}{12}$

Events are dependent when their probabilities do affect each other. In this context conditional probabilities are defined and calculated. For example, the probability of drawing a heart from a complete card deck is *25%*. If the second card is drawn without reinserting the first card, the probability of that card being a heart is lower since one card has already been removed. Therefore, the probability of drawing the second card is dependent on the first one. Conditional probability is denoted as *P(Y | X)* , which is read as "the probability that *Y* occurs given that *X* has occurred". Conditional probability is formally defined as

$$P(Y \mid X) \mathrm{X} \frac{P(X \quad Y)}{P(X)} \tag{2}$$

If the events *X* and *Y* are independent, then the conditional probability for *Y* given that *X* has occurred is equal to the probability of *Y*. This result can be easily obtained by substituting (1) into (2).

$$P(Y \mid X) \mathrm{X} \frac{P(X \quad Y)}{P(X)} \mathrm{X} \frac{P(X).P(Y)}{P(X)} \mathrm{X} P(Y) \tag{3}$$

The unconditional (prior) probability of an event *X*, *P(X)*, is the probability of the event before any evidence is presented. The evidence is the perception that affects the degree of belief in an event. The conditional probability of an event is the probability of the event after the evidence is presented.

For example, the event that a person has anti-virus program can be event *V* and the event that a person has a spam-filter event *S*. If there is evidence that *60%* of all people have an anti-virus program and that *20%* of all people have a spam-filter and an anti-virus program, then the probability of a person having a spam-filter given that he/she has an anti-virus program can be calculated as follows.

$$P(S \mid V) \mathrm{X} \frac{P(S \quad V)}{P(V)} \mathrm{X} \frac{0.20}{0.60} \mathrm{X} \frac{1}{3}$$

The spam classification in NB is based upon Bayes theorem that defines the relationship between the conditional probabilities of two events.

4.4 Bayes Theorem

Bayes theorem provides a way to calculate the probability of a hypothesis, here the event *Y* , given the observed training data, here represented as *X* :

$$P(Y \mid X) \mathrm{X} \frac{P(X \mid Y)P(Y)}{P(X)} \tag{4}$$

This simple formula has enormous practical importance in many applications. It is often easier to calculate the probabilities, *P(X | Y), P(Y), P(X)* when it is the probability *P(Y | X)* that is required. This theorem is central to Bayesian statistics, which calculates the probability of a new event on the basis of earlier probability estimates derived from empirical data.

4.5 Classical vs. Bayesian Statistics

4.5.1 Using Statistics

Statistics is used to draw conclusions from data and to predict the future in order to answer research questions such as “Is there a relationship between a student’s IQ and height?” To answer such questions students’ IQ and height data must be collected. This can be achieved by performing experiments. For example, an IQ-test is given and the height of each student is recorded. A plot is made of IQ v Height and it is then possible to detect whether or not a correlation exists. Statistical tests can be applied to answer research questions, to confirm or reject certain hypothesis. There are two essential statistical methods, classical (or related to frequency) (*Hinton R. 2004*) and Bayesian (*Bullard F. 2001 & Heckerman D. 1995 & Lee P. 2004*).

4.5.2 Objective and Subjective Probabilities

In classical statistics all attention is devoted to the observed data, the frequencies which are generally collected from repeated trials. For example, in the case where the research question consists of deciding whether a particular die is biased or not, multiple rolls are required to obtain sufficient data. The data is then used as evidence to determine whether the observed results are significantly different to the expected for a non-biased die. This can be used as evidence that a die is biased.

As opposed to classical, Bayesian statistics takes a subjective degree of belief into account, the prior data. With Bayesian statistics it is possible to take subjective probabilities together with the collected data to obtain the probability of the die being biased.

4.5.3 Inference Differences

Another difference between classical and Bayesian statistics is how their inference is performed. In classical statistics an initial assumption or the hypothesis about the research question is first made. It is usually called a null hypothesis. A single or several alternative hypotheses can also be defined. Then the relevant evidence or data are collected. This evidence measures how different the observed results are from the expected if the null hypothesis was true. The measurement is given in terms of a

calculated probability called the *p-value*. It is the probability of obtaining the observation found in the collected data, or other observations which are even more extreme.

The significance level is the degree of certainty that is required in order to reject the null hypothesis in favor of the alternative. A typical significance level of *5%* is usually used. The notation is $\alpha=0.05$. For this significance level, the probability of incorrectly rejecting the null hypothesis when it is actually true is *0.05*. If higher protection is needed, a lower α can be selected. Once the significance level is determined and the *p-value* is calculated the following conclusion is drawn. If the probability of observing the actual data under the null hypothesis is small ($p < \alpha$) the null hypothesis is not true and it can be rejected. This means that the alternative hypothesis is accepted. The converse is not true. If the *p-value* is big ($p > \alpha$), then there is insufficient evidence to reject the null hypotheses.

For example, let the null hypothesis, “the die is unbiased” be assumed to be true. If the die is rolled many times and *70%* of all outcomes are sixes, the statistical test will calculate the probability of obtaining a six in *70%* of outcomes or higher (*p-value*). The probability distribution for the outcomes observed using an unbiased die is used for this calculation. If the *p-value* is lower than *0.05* then according to classical statistics the null hypothesis can be rejected and the die will be considered to be biased.

In Bayesian statistics, however, a probability is really an estimate of a belief in a particular hypothesis. The belief that a six occurs once in every six rolls of the die comes from both, prior considerations about fair die and the empirical results that have been observed in the past. Bayesian statistics evaluates the probability of a six by taking the previous data collected into consideration. For many researchers this approach is more intuitive than the inference of classical statistics.

4.5.4 Example of Statistical Spam Classification

In order to implement either the classical or the Bayesian statistics to classify an e-mail message as spam, some data must be available. This usually occurs as a collection of e-mail messages labeled as spam or non-spam and are referred as a corpus (training data).

In addition, the information about the frequency distribution of the tokens in the corpus is available. This means that every token is accompanied by the number of times.

4.5.4.1 Classical Statistics

When testing a new e-mail message the starting point is the null hypothesis stated as "the message is spam". The alternative hypothesis is "the message is non-spam". The significance level is chosen to be *5%* or $\alpha = 0.05$. To classify a message as spam a frequency distribution of its tokens is created and compared to the previous training data (spam corpus) with an appropriate statistical test (¡ ^{2} tests can be used to analyze frequency data). The statistical test will give a probability (*p -value*) and if it is lower than the significance level the null hypothesis is rejected and the message is concluded not to be spam. Otherwise the null hypothesis is accepted.

4.5.4.2 Bayesian Statistics

Bayesian probabilistic reasoning has been used in machine learning since the 1960s, especially in medical diagnosis. It was not until 1998 that Sahami *et al*. 1998 applied a Bayesian approach to classify spam. With Bayesian statistics the probability of a model based on the data is calculated as opposed to classical statistics which calculates the probability of the data given a hypothesis (model). To illustrate this, consider the previous example where classical statistics was used to verify the null hypothesis (the message being spam), either the message is classified as spam or not. Bayesian statistics calculates the probability of a message being spam. For example, Bayesian statistics can calculate that a message has an *82%* chance of being a spam.

Chapter 5 NAIVE BAYESIAN SPAM FILTERING

5.1 The model

A general Naive Bayesian spam filtering can be conceptualized into the model as presented in the figure below. It consists of four major modules, each responsible for four different processes: message tokenization, probability estimation, feature selection and Naive Bayesian classification.

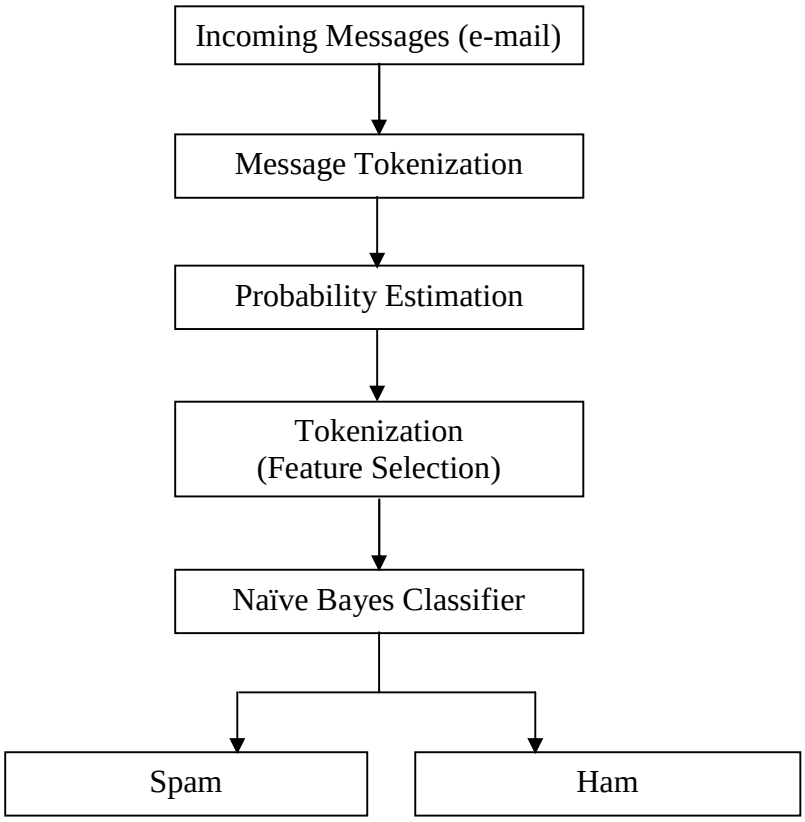

Figure 5.1: A model of Naïve Bayesian Spam Filtering

When a message arrives, it is firstly tokenized into a set of features (tokens), F . Every feature is assigned an estimated probability that indicates its spaminess. To reduce the dimensionality of the feature vector, a feature selection algorithm is applied to output a subset of the features, F^1 F . The Naive Bayesian classifier combines the probabilities of every feature in F^1 , and estimates the probability of the message being spam.

5.2 Naive Bayesian Classifier

In terms of a spam classifier Bayes theorem (4) can be expressed as

$$P(C \mid F) = \frac{P(F \mid C)P(C)}{P(F)} \tag{5}$$

where $F=\{f_1, \ldots, f_n\}$ is a set of features and $C=\{ham, spam\}$ are the two classes. Where the number of features, n, is large, computing $P(F \mid C)$ can be time consuming. Alternatively, it can be assumed that the features, which are usually words appearing in the e-mail message, are independent of each other.

Using the assumption for independence, according to (1), the joint probability for all n features can be obtained as a product of the total individual probabilities

$$P(F \mid C) = \prod_{i=1}^{n} P(f_i \mid C) \tag{6}$$

Inserting (6) into (5) yields

$$P(C \mid F) = \frac{P(C) \prod_{i=1}^{n} P(f_i \mid C)}{P(F)} \tag{7}$$

The denominator $P(F)$ is the probability of observing the features in any message and can be expressed as

$$P(F) = \sum_{k=1}^{m} P(C_k) . \prod_{i=1}^{n} P(f_i \mid C_k) \tag{8}$$

Inserting (8) into (7) the formula used by the Naive Bayesian Classifier is obtained

$$P(C \mid F) = \frac{P(C) \prod_{i=1}^{n} P(f_i \mid C)}{\sum_{k=1}^{m} P(C_k) . \prod_{i=1}^{n} P(f_i \mid C_k)} \tag{9}$$

If *C = spam* then (9) can basically be read as: "The probability of a message being spam given its features equals the probability of any message being spam multiplied by the probability of the features co-occurring in a spam divided by the probability of observing the features in any message".

To determine whether or not a message is spam the probability given by (9) is compared to a threshold value, $t \, \mathrm{X} \frac{\mathrm{a}}{1 \Gamma \mathrm{a}}$. If *P(C = spam | F) > t* then the message is classified as spam. For example, when $^{\mathrm{a}}$ *= 9* then *t = 0.9* meaning that blocking one legitimate message is of the same order as allowing *9* spam messages to pass.

Applying the Bayesian theorem on spam categorization, *Graham (2000)* and *Robinson (2003)* used the following formula:

$$P(w) \, \mathrm{X} \frac{b(w)}{b(w) \, \Gamma \, g(w)}$$

Where *b(w)* is the number of spam emails containing the word *w* divided by the total number of spam emails and *g(w)* is the number of ham emails containing the word *w* divided by the total number of ham emails. *p(w)* is the probability that a randomly chosen email containing the word *w* will be spam. Switching *b(w)* and *g(w)* in the above formula gives the probability that a randomly chosen email containing the word *w* is ham.

Robinson (2003) noted that dealing with the more than one time occurring words in an email requires the calculation of the degree of belief *f(w)*. The degree of belief *f(w)* is calculated as follows:

$$f(w) \, \mathrm{X} \frac{(s \mid x) \, \Gamma \, (n \mid p(w))}{s \, \Gamma \, n}$$

Where *s* is the strength of background information that means repetition of any token, *x* is the assumed probability based on general background information that means probability of any token to be a spam. The *n* is the number of emails received which contains the word *w*. So *f(w)* is used instead of *p(w)* for calculating the probabilities as noted by Robinson (2003) for more reliable classification.

An inverse chi-square distribution with 2n degrees of freedom is used to combine the probabilities of every word w. The combined probability of ham was calculated as follows:

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

Where C^{-1} is the inverse chi-square function used to derive the probability from a chi-square-distributed random variable. Similarly, the combined probability of spam *S* is calculated as follows:

$$S = C^{-1}(-2\ln \prod_{w} 1 - f(w), 2n)$$

After calculating both *S* and *H*, the value of *I* is calculated which indicates whether the evidence is in-favor of concluding that an email is spam or ham. When *I* is near 1, the evidence provided indicates that the message is spam and when it is near 0 the evidence indicates that the message is ham. *I* is calculated using the following formula:

$$I = \frac{1 + H - S}{2}$$

Where *I* is the indicator whether the message is ham or spam, *H* is the combined probability of ham, and *S* is the combined probability of spam *(Robinson, G., 2003).*

5.3 χ^2 Statistics

The χ^2 test of independence compares patterns of observed with expected frequencies to determine whether or not they differ significantly from each other. As the χ^2 distribution is continuous and the observed frequencies are discrete the χ^2 test is not reliable for small cell values. Recommendations are to keep frequencies larger than five. The chi square statistics is defined as

$$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Where *O* is the observed and *E* is the expected frequency calculated from the joint corpus. The degrees of freedom are defined from the size of the contingency table as $df = (I-1)(J-1)$. Here is a two way contingency table for a term *t*

	t	$\bar{t}$
$c = good$	$A = freq(t, c)$	$C = freq(\bar{t}, c)$
$c \neq good$	$B = freq(t, \bar{c})$	$D = freq(\bar{t}, \bar{c})$

The value of $¡^2$ can be calculated as follows:-

$$¡^2 \text{X} \frac{N.(AD \text{Z} CB)^2}{(A \Gamma C).(B \Gamma D).(A \Gamma B).(C \Gamma D)} 2$$

where $N = A + B + C + D$. The degree of freedom is $1 = (2 - 1) \times (2 - 1)$. The $¡^2$ value is zero if t is completely independent of c. Following on from the work of (*Yang & Pedersen 1997*) this implementation uses the maximum $¡^2$ value between the feature and class as the weight.

$$¡^2_{\max} \text{X} \arg\max_i \sum ¡^2(t, c_i)*$$

Chapter 6 IMPLEMENTATION

6.1 Tokenization

Most of real-world application of the Naive Bayesian uses word-based tokenization. Tokenization is the activity of constructing a vector representing the content of email. Message tokenization is the process of tokenizing an e-mail message using carefully selected delimiters. The constructed tokens are considered the vector attributes of messages and their frequency ratios are used in the calculation of probabilities during classification. Deriving tokens from messages is a multi-stage preprocessing task in which text is filtered according to specified criteria and grouped into a dataset accessible to the classification function. Two schemes of tokenization were implemented. A word-distribution based tokenization in which every word is considered a token and a trigram based tokenization in which every 3-word phrase is considered a token.

6.1.1 Word-based Tokenization

Tokenization of the email body based on word distribution is simply done by collecting words that are separated by white-space – whether the white-space is a space, a tab, a carriage return, a new-line character, a combination of white-space characters, or multiple white-space characters. Prior to splitting the body of a message, several preprocessing steps are taken to filter out the irrelevant content which mostly consists of non-alphanumeric characters. Almost all non-alphanumeric characters are replaced by a space character. This in effect removes any non-alphanumeric characters from the body of the email. Some non-alphanumeric characters are kept such as “$”, “!”, “-”, and “%”. These are kept in the body of emails because of their high association with words and numbers used in spam email.

The implemented tokenizer deals with HTML as well as plain-text content in much the same way. This means that HTML tags are stripped and URLs are torn apart into words.

Graham (2002) noted that words that are less than 3 characters in length carry little to no statistical significance. So, the research work has ignored words that are less than 3 characters in length. Moreover, words that are longer than 12 characters were

similarly of little statistical significance and are ignored. *Robinson (2003)* and the SpamBayes project (*Meyer & Whateley, 2004*) concurred with Graham's conclusion. Additionally, words are automatically converted to lower-case prior to tokenization. This allows words that are capitalized to add to the significance of non-capitalized words rather than treating them as separate tokens with their own frequency ratios. This makes sense because capitalization is merely a grammatical structure that carries no syntactic information.

6.1.2 Trigram-based Tokenization

Trigram-based tokenization follows the same preprocessing steps as in the word-based tokenizer. This process includes separating words based on white-space, stripping non-alphanumeric characters, including the "$", "!", "-", and "%" characters, stripping HTML encoded messages and converting words to lower-case. In this method every three-word phrase has considered as a token.

6.2 Datasets

Emails used in testing are collected from the public corpus of *SpamAssassin*[16]. The public corpus is pre-categorized into three categories easy_ham, hard_ham, and spam. The easy_ham and the spam corpora are only used. After examining the emails contained in the hard_ham, it is found that most of the messages originated from newsletters and product advertisement related emails that had little to no relevancy to the mailing list of SpamAssassin which is the largest contributor to the corpus. The total amount of emails used in the training corpus and in testing the classification is summarized in the following table:

Table 6.2.1: The amount of emails used in training and testing the classification

Corpus	Number of messages	Distribution
Spam	300	33.33%
Ham	600	66.66%
Total	900	100%

The emails are collected from the corpus of year 2010. The accuracy of classifier depends on the amount of messages processed by the trainer. During analysis, an equal

[16] See http://spamassassin.apache.org/publiccorpus/

ratio of spam to ham was used in training both the word-based classifier and the trigram-based classifier. The following table describes the distribution of messages used in the training corpus:

Table 6.2.2: The distribution of spam to ham in the training corpus during result analysis

Corpus	Number of messages	Distribution
Spam	150	33.33%
Ham	300	66.66%
Total	450	100%

6.3 Training and Classification

Training the Naive Bayesian classifier is the one of the most fundamental and crucial tasks in implementing a classifier. Training comes only after the application is able to fully tokenize and deal with the textual contents of emails. Rather than storing pre-calculated probabilities to each token stored in the training corpus, only the number of occurrences of each token is stored. The classifier is tested by incrementing the number of messages or emails into training corpus.

Every token is taken into account during probability estimation. Tokens are given a frequency of one regardless of the number of occurrences in a single email. This is deliberately done to correctly calculate the probabilities according to the above mentioned formulas. Every token is searched in the training corpus. If the token was found in the training corpus, f(w) is calculated whether it belongs to the spam, ham, or unsure. However, when the token does not exist, the presumed probability with the value 0.5 (neutral) is given to the token. When all tokens have been considered, all calculations are temporarily stored and the value of I value is calculated using the formulas described above.

By the trial and error method, it is found that ±1 σ from the presumed probability 0.5 is the optimal cutoff point whereby when a value less than -1 σ (%31.7) was returned, the email was identified as ham and when a value more than +1 σ (%68.2) was returned, the email was identified as spam. For any message that returns a value that lies between ±1 σ, an “unsure” classification was returned. *Graham (2002)* classifier used two

categories spam and ham. However, as noted by *Robinson (2003)* this leads to falsely assuming that an email that only marginally falls below the cutoff point to be regarded as ham when in reality a more correct term describing the email would be “I do not have enough evidence to classify this email” or “unsure” for short. This allows us to distinguish failure of classification from success in classifying ham emails. Additionally, this means that rather than setting a single probability threshold for classifying spam, two thresholds needs to be set – an upper threshold for classifying spam and a lower threshold for classifying ham. Extra care needs to be taken when filtering spam due to the potential disastrous effects of falsely identifying a legitimate email as spam. When the value of I drops below 68.2% (or +1 σ), non-action should be taken by the filter irrespective. Although such approach might yield higher occurrences of false negatives in the users’ inboxes, it is indeed – as noted by many – better than the detrimental effect of producing a false positive result. The usability of spam filters are greatly affected by false positive results but false negative effects the result negligible. The cutoff point should be decreased if a large amount of false negative results were observed. Similarly, when a false-positive was identified, the suggested course of action is to increase the cutoff point.

Chapter 7 EVALUATION TECHENIQUES

The evaluation of a classifier depends on the domain in which it is implemented. The contents of trained and tested emails, size of trained and tested corpus are also the affecting factors for evaluation of the classifier. Although, Precision and Recall rates are the formal standard measuring tools for evaluation of spam classification (*Sahami et al. 1998*).

7.1 Precision and Recall

Recall is the proportion of relevant items that are retrieved, which in this case is the proportion of spam messages that are actually recognized.

Precision is defined as the proportion of items retrieved that are relevant. In the spam classification context, precision is the proportion of the spam messages classified as spam over the total number of messages classified as spam. Thus if only spam messages are classified as spam then the precision is 1. As soon as a good legitimate message is classified as spam, the precision will drop below 1.

The recall *(r)* and precision *(p)* rates were calculated using the following formulae:

$$recall = \frac{n_{spam|\ spam}}{n_{spam|\ spam} + n_{spam|\ ham}} \qquad precision = \frac{n_{spam|\ spam}}{n_{spam|\ spam} + n_{ham|\ spam}}$$

where

$n_{spam->spam}$ is the total number of messages accurately identified as spam

$n_{spam->ham}$ is the total number of spam messages identified as ham (false-negative), and

$n_{ham->spam}$ is the total number of legitimate messages marked as spam (false-positive).

The precision calculates the occurrence of *false positives* which are good messages classified as spam. When this happens *p* drops below *1*. Such misclassification could be a disaster for the user whereas the only impact of a low recall rate is to receive spam messages in the inbox. Hence it is more important for the precision to be at a high level than the recall rate. The precision and recall reveal little unless used together.

Commercial spam filters sometimes claim that they have an incredibly high precision value of *0.9999%* without mentioning the related recall rate. This can appear to be very good to the untrained eye. A reasonably good spam classifier should have precision very close to *1* and a recall rate > *0.8*.

Classification in previous studies used two classes spam and ham. When an email fails to accumulate enough probability to overcome a set threshold, it is automatically rejected and classified as ham. According to the filter devised by *Robinson (2003)* and later used by the *SpamBayes* team, an "Unsure" category was added to distinguish failure to classify an email from classifying an email as ham. However, to adjust the above formulas to take into account the third category "unsure", the number of "unsure" classifications was added to the $n_{spam->ham}$ value gives a recall rate comparable to previous studies using the same performance measure. The recall and precision rates were calculated in both the word-based spam filter test run and the trigram-based spam filter test run. The same collection of emails was used in training the classifier in both instances. Moreover, the collection of emails used in testing was the same. The recall and precision rates were calculated every incremental step during which emails were added to the training corpus.

Chapter 8 RESULTS AND ANALYSIS

Data retrieved from the test of the word-based spam filter showed improved recall rates whenever new emails were added to the training corpus. Both of the spam filters have shown the better results as increase in training. Word-based spam filter has shown better result than trigram-based spam filter on the basis of recall rates. Trigram-based spam filter has shown better result than word-based spam filter on basis of precision rates. Hence, word-based filter is good for small data set but Tri-gram based filter is good for large data set because as increase in training and data set the recall rates and precision of trigram is continuously increasing. But in case of word-based spam filter, it has shown steady at the end of testing.

The results of both word-based and trigram-based spam filter is shown as follows:-

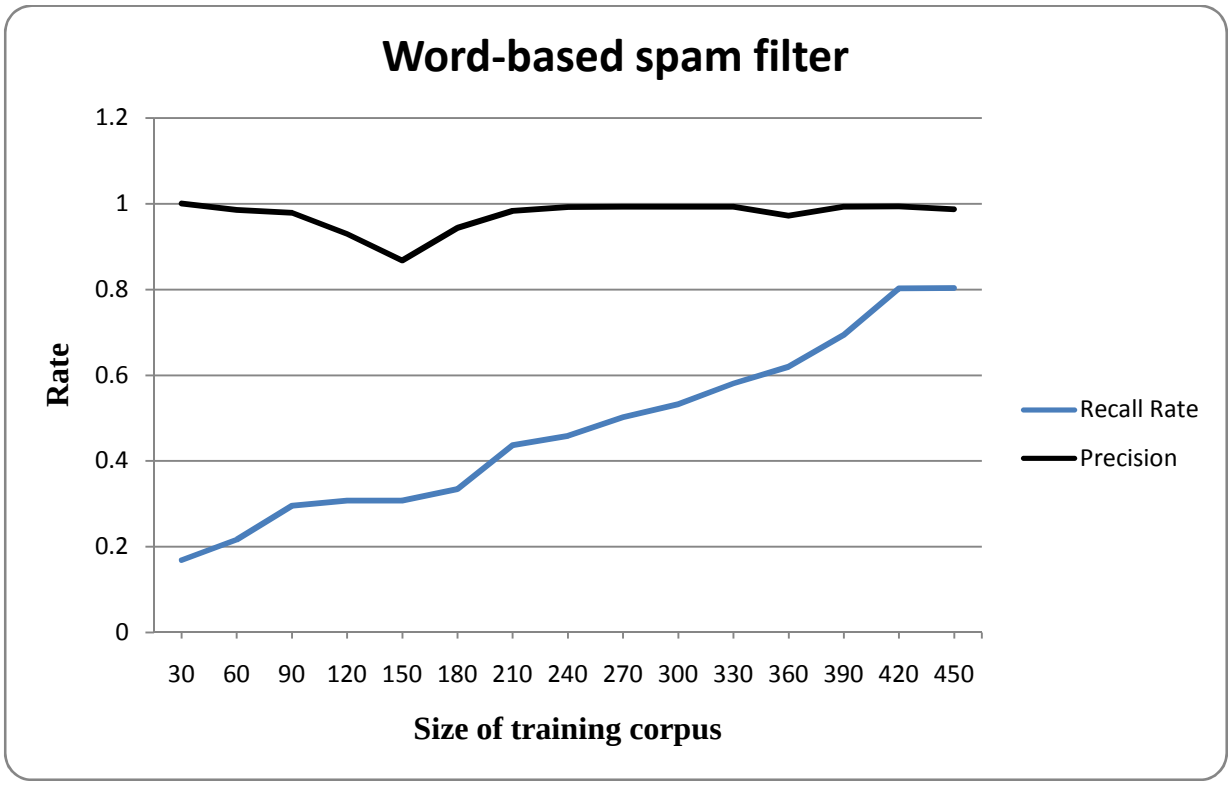

Figure 8.1: Recall and precision rates of word-based spam filter

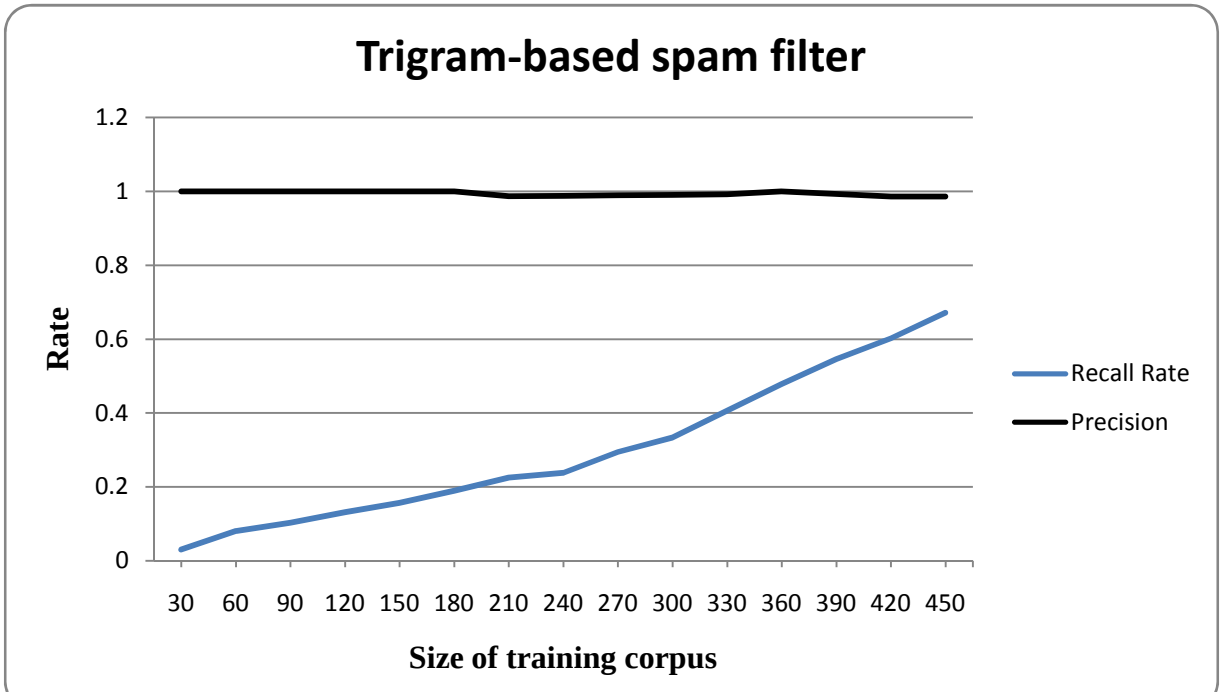

Figure 8.2: Recall and precision rates of trigrim-based spam filter

Comparative summary statistics have been computed between the data obtained from the test on the word-based spam filter and the test on the trigram-based spam filter which are summarized in the following tables:

Table 8.1: Summary statistics for the recall and precision rates

	Word-based	**Trigram-based**
Mean Recall	0.470584	0.299328
Max Recall	0.8083371	0.671498
Mean Precision	0.973271	0.993947
Max Precision	1.0	1.0

Chapter 9 CONCLUSION AND FUTURE WORK

9.1 Conclusion

The main concern for this study was to examine the efficacy of word-based Bayesian spam filters as opposed to trigram-based Bayesian spam filters. The comparison of efficiency between these spam filter has done on the basis of the precision and recall.

The pattern of improvement in the recall rate differs in the two spam filters. Whilst the pattern in the word-based spam filter was of a rapid increase followed by stability, the pattern observed in the trigram-based spam filter was of a gradual increase. The maximum recall rate achieved by the word-based spam filter succeeded the rate achieved by the trigram-based spam filter.

On the basis of results shown by the testing, word-based spam filter is better for low volume of datasets but for a large datasets trigram-based spam filtering will be more suitable due to its higher reliability and accuracy.

For the small organizations or personal users who deal with a small amount of emails, word-based spam filter is better for implementation but for large organizations, trigram-based spam filter is suitable. The spam filter can be used as per need by any organizations and persons for their email management system in mail server. It can be used for document classification containing text rather than graphics and pictures. These are the major contributions of this research work.

9.2 Future Work

The Naive Bayesian spam filter can be improved, make more accurate and reliable in future as follows:-

- An efficient algorithm can be developed by combining with other algorithms and theories like Genetic Algorithm, Automata Theory, Association Rules, Meta-Learning approaches such as EM (Expectation Maximization) and Boosting.

- The word-position-based attributes can be implemented for Naive Bayesian spam filter.
- Different techniques can be incorporated with Naive Bayesian spam filter like different ways of feature selection such as filters and wrapper induction method, classification using adaptive ontology, term weighting, cross validation, probability estimation by using statistical tools like Absolute estimator, Laplace estimator, Lidstone Estimate.

References

[1] Androutsopoulos I., Paliouras G., Karkaletsis V., Sakkis G., Spyropoulos C. and Stamatopoulos, P. (2000a) Learning to filter spam email: A comparison of a naive bayesian and a memory-based approach. In *Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2000)*.

[2] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, George and Spyropoulos, C.D. (2000b), An Evaluation of Naive Bayesian Anti-Spam Filtering. In Potamias, G., Moustakis, V. and van Someren, M. (Eds.), *Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000)*, Barcelona, Spain, pp. 9-17.

[3] Androutsopoulos, I., Paliouras, G., Michelakis, E. (2004), Learning to Filter Unsolicited Commercial E-Mail. Athens University of Economics and Business and National Centre for Scientific Research "Demokritos" Bevilacqua-Linn M. (2003), Machine Learning for Naive Bayesian Spam Filter Tokenization Breiman, L., and Spector, P. (1992), Submodel selection and evaluation in regression: The Xrandom case. International Statistical Review, 60, 291-319.

[4] Bullard, F. (2001), A Brief Introduction to Bayesian Statistics, http://courses.ncssm.edu/math/TALKS/PDFS/BullardNCTM2001.pdf, 2011-10-13.

[5] Carreras, X., & Marquez, L., (2001). 'Boosting Trees for Anti-Spam Email Filtering'. *Proceedings of RANLP-2001*. September 5-7, 2001, Tzigov Chark, Bulgaria, 58-64.

[6] Chen D., Chen T. and Ming H. Spam Email Filter, Using Naive Bayesian, Decision Tree, Neural Network, and AdaBoost

[7] Domingos, P., and Pazzani, M. (1996), Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. *Proceedings of the 13th Int. Conference on Machine Learning, pp. 105–112,* Bari, Italy.

[8] Drucker H., Wu D., and Vapnik, V.N., Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048–1054, Sep. 1999.

[9] Forman G. (2003), An Extensive Empirical Study of Feature Selection Metrics for Text Classification. JMLR 3(Mar):1289-1305.

[10] Graham, P. (2002), A plan for spam. http://www.paulgraham.com/spam.html, 2011-10-13

[11] Hinton, R. (2004), Statistics Explained, 2nd Edition, Routledge

[12] Holden, S. (2003), Spam Filters, http://freshmeat.net/articles/view/964/, 2011-10-18.

[13] Johnson, D and Malhotra, V and Vamplew, P (2006) *More Effective Web Search Using Bigrams and Trigrams.* Webology, 3 (4). p. 35. ISSN 1735-188X

[14] Kanich, Kreibich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., et al. (2008*). Spamalytics: an empirical analysis of spam marketing conversion.* Paper presented at the Proceedings of the 15th ACM conference on Computer and communications security.

[15] Lai, C., & Tsai, M., (2004). 'An Empirical Performance Comparison of Machine Learning Methods for Spam E-mail Categorization'. *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems.* December 5-8, 2004, Kitakyushu, Japan.

[16] Lee, P. (2004), Bayesian Statistics an introduction, Third Edition, Provost of Wentworth Collage, University of York, UK. Arnold Publishers.

[17] Lyon, C., Barrett, R., & Malcolm, J. (2006). Plagiarism Is Easy, But Also Easy To Detect. *Plagiary: Cross-Disciplinary Studies in Plagiarism, Fabrication, and Falsification,* 1(5): 1-10

[18] Mertz, D., (2002), Spam filtering techniques, http://www-106.ibm.com/developerworks/linux/library/l-spamf.html#author1, 2011-10-12.

[19] Meyer, T., & Whateley, B., (2004). 'SpamBayes: Effective open-source, Bayesian based, email classification system'. *Proceedings of the First Conference on Email and Anti-Spam.* July 30-31, 2004.Mountain View, California, USA.

[20] Prakash, Vipul Ved. (2006), Vipul's Razor, http://razor.sourceforge.net/, 2011-10-13

[21] Rainie, L., & Fallows, D. (2004), *The impact of CAN-SPAM legislation*. Washington D.C: Pew/Internet

[22] Robinson, G. (2003). 'A Statistical Approach to Spam Filtering'. *Linux Journal* [online]. 1st March. Accessed on 4th November 2008. Available at: http://www.linuxjournal.com/article/6467

[23] Sahami, M., Dumais S., Heckerman D., and Horvitz, E. (1998), A Bayesian approach to filtering junk e-mail. Learning for Text Categorization. *Papers from the AAAI Workshop, Madison Wisconsin, pp. 55–62. AAAI Technical Report WS-98-05*.

[24] Salton, G., & McGill, M., (1983). *Introduction to Modern Information Retrieval.* McGraw-Hill: New York, NY, USA.

[25] SpamAssassin Public Corpus, http://spamassassin.apache.org/publiccorpus/, 2010

[26] Wiener E., Pedersen, J.O. and Weigend, A.S. (1995) A neural network approach to topic spotting. *In Processings of the Fourth Annual Symposium on Document Analysis and Information Retrieval* (SDAIR'95).

[27] Yang Y. and Liu X. (1998), A re-examination of text categorization methods. *School of Computer Science, Carnegie Mellon University*

[28] Yang, Y. and Pedersen, J.O. (1997), A Comparative Study on Feature Selection in Text Categorization. *Proceedings of ICML-97, 14th International Conference on Machine Learning.*

[29] Youn, Seongwook and McLeod, Dennis (2006), "A Comparative Study for Spam Mail Classification", *2nd International Conference on Systems, Computing Sciences, and Software Engineering (SCS2 '06)*, USA.

[30] Zipf, G. (1949), *Human Behaviour and the Principle of Least Effort,* Addison Wesley, Cambridge.

Appendix A

Sample emails for spam training

Sample email 1:

```
<strong>Viagra for sale in united states....</strong>\n\nViagra
for sale in united states. By mail sale viagra. Viagra sale.
Viagra for sale best online pharmacy....'
```

Sample email 2:

```
<strong>No prescription needed vicodin....</strong>\n\nVicodin.
Buy vicodin online....
```

Sample emails for ham training

Sample email 1:

```
[...] the articles of How to manage thousands visitors, right now
I write about the SQLite emblemed database [...]
```

Sample email 2:

```
The class only read and write RSS2.\r\n\r\nI don''t know what you
mean with "multidemtional arrays". I''ve try with workpress, and
blogger, and works ok.\r\n\r\nThank you for write, and if have
any doubt feel free to ask free support on the php-classes site,
or post here.\r\n\r\nbest regards
```

Sample email 3:

```
hi,\r\n\r\nthanks for mentioning my class! I just have released a
minor bugfix (0.1.2), which was necessary due to recent changes
in the YouTube Upload-forms. get more info
at:\r\n\r\nhttp://nonsmokingarea.com/blog/2007/07/01/youtube-
google-maps-phptube-012/
```

Sample email 4:

```
New bugs were fixed!!!\r\n\r\nThe new version is here
http://www.phpclasses.org/dnserver
```

Sample emails for testing

Sample email 1:

```
buy imitrex tramadol <a href="http://buytramadol1.beeplog.de"
rel="nofollow">tramadol buy insurance</a> Viagra and vicodin… No
prescription needed!
```

Sample email 2:

```
To know more about AJAX and its usage, you can visit my blog
at:\r\nhttp://rochakchauhan.com/blog/category/programming/ajax/.
You can also get tutorials of SQLite and RSS2.
```

Sample email 3:

```
So can you add a custom text on web pages using this class?  It
sounds good :).
```

Appendix B

Results from word-based spam filter test

S_S	S_H	H_S	H_H	Unsure	Spam	Ham	Total	Recall	Precision
58	0	0	106	286	10	20	30	0.168605	1
67	2	1	139	241	20	40	60	0.216129	0.985294
89	3	2	147	209	30	60	90	0.295681	0.978022
92	3	7	144	204	40	80	120	0.307692	0.929293
98	2	15	116	219	50	100	150	0.30721	0.867257
100	2	6	145	197	60	120	180	0.334448	0.943396
117	1	2	180	150	70	140	210	0.436567	0.983193
120	1	1	187	141	80	160	240	0.458015	0.991736
129	2	1	192	126	90	180	270	0.501946	0.992308
132	2	1	201	114	100	200	300	0.532258	0.992481
133	2	1	220	94	110	220	330	0.580786	0.992537
137	2	4	225	82	120	240	360	0.61991	0.971631
136	1	1	253	59	130	260	390	0.693878	0.992701
142	1	1	272	34	140	280	420	0.80226	0.993007
143	1	2	270	34	150	300	450	0.803371	0.986207

Appendix C

Results from trigram-based spam filter test

S_S	S_H	H_S	H_H	Unsure	Spam	Ham	Total	Recall	Precision
13	0	0	22	415	10	20	30	0.030374	1
33	3	0	41	373	20	40	60	0.080685	1
40	4	0	63	343	30	60	90	0.103359	1
49	4	0	78	319	40	80	120	0.13172	1
58	3	0	81	308	50	100	150	0.157182	1
68	3	0	92	287	60	120	180	0.189944	1
74	3	1	121	251	70	140	210	0.22561	0.986667
77	3	1	126	243	80	160	240	0.23839	0.987179
93	2	1	133	221	90	180	270	0.294304	0.989362
100	2	1	149	198	100	200	300	0.333333	0.990099
119	2	1	156	172	110	220	330	0.406143	0.991667
134	2	0	170	144	120	240	360	0.478571	1
136	1	1	200	112	130	260	390	0.546185	0.992701
138	1	2	219	90	140	280	420	0.60262	0.985714
139	0	2	241	68	150	300	450	0.671498	0.985816

Appendix D

Trigram Module

```
class ngram {

   var $text;
   var $length;
   var $ngrams;

   function ngram($letter=1) {
      $this->setLength($letter);
   }

   function setLength($length=1){
      $this->length=$length;
   }

   function setText($text) {
      $this->text =" ".$text."  ";
   }

   function setInitialNgram($arg) {
      $this->ngrams = $arg;
   }

   function getnGrams() {
      return $this->ngrams;
   }

   function extract() {
      $txt = & $this->text;
      $len = strlen($txt);
      $length = & $this->length;
      $ngrams = & $this->ngrams;
      $buf='';
      $ultimo='';
      for($i=0; $i < $len; $i++) {
         if ( strlen($buf) < $length) {
            if ( !useful($txt[$i]) )
               continue;

            if (is_space($txt[$i]) && is_space($ultimo))
                continue;

            $buf .= is_space($txt[$i]) ? '_' : $txt[$i];
            $ultimo = $txt[$i];
         } else {
            $buf = strtolower($buf);
```

```
                $buf = str_replace(" ","_",$buf);
                $ngrams[$buf] = isset($ngrams[$buf]) ? $ngrams[$buf] + 1 : 1;
                $ultimo=";
                $buf = ";
                $i--;
            }
        }
    }
}
```

Appendix E

Training Module

```
class trainer {
   var $examples;
   var $ngram;
   var $knowledge;

   function trainer() {
      $this->ngram = new ngram;
   }
  function add_example($text, $clasification) {
      $this->examples[$clasification][] = $text;
   }
   function setPreviousLearn($f) {
      $this->previous = $f;
   }
  function extractPatterns() {
      $previous = & $this->previous;
      $examples = & $this->examples;
      $ngram = & $this->ngram;
      $knowledge = & $this->knowledge;

      foreach($examples as $tipo => $texts) {
         $params[$tipo] = 0;
         $ngram->setInitialNgram( isSet($previous[$tipo]) ? $previous[$tipo] : array() );
         foreach ($texts as $text) {
            $ngram->setText($text);
            for($i=3; $i <= 5;$i++) {
               $ngram->setLength($i);
               $ngram->extract();
            }
         }

         $actual = & $knowledge[$tipo];
         foreach( $ngram->getnGrams() as $k => $v) {
            $actual[$k]['cant'] = $v;
            $params[$tipo] += $v;
         }
      }
      $this->computeBayesianFiltering($params);
   }
```

```
   function computeBayesianFiltering($param) {
      $knowledge = & $this->knowledge;
      foreach($knowledge as $tipo => $caracterist) {
         foreach($caracterist as $k => $v) {
            $t = ($v['cant']/$param[$tipo]);
            $f = 0;
            foreach($param as $k1 => $v1)
               if ( $k1 != $tipo) {

                  $f += isset($knowledge[$k1][$k]['cant']) ? $knowledge[$k1][$k]['cant'] /
$v1 : 0;
               }
            $knowledge[$tipo][$k]['bayesian'] = $t / ($t + $f);
         }
      }
   }
}
```

Appendix F

Spam checking Module

```
function isItSpam_v2($text,$type) {
      $ngram = new ngram;
      $ngram->setText($text);

      for($i=3; $i <= 5;$i++) {
         $ngram->setLength($i);
         $ngram->extract();
      }

      $fnc = $this->_source;
      $ngrams =  $ngram->getnGrams();
      $knowledge =  $fnc( $ngrams,$type );
      $total=0;
      $acc=0;

      $N = 0;
      $H = $S = 1;

      foreach($ngrams as $k => $v) {
         if ( !isset($knowledge[$k]) ) continue;
         $N++;
         $value = $knowledge[$k] * $v;
         $H *= $value;
         $S *= (float)( 1 - ( ($value>=1) ? 0.99 : $value) );
      }

      $H = $this->chi2Q( -2 * log( $N *  $H), 2 * $N);
      $S = (float)$this->chi2Q( -2 * log( $N *  $S), 2 * $N);
      $percent = (( 1 + $H - $S ) / 2) * 100;
      return is_finite($percent) ? $percent : 100;
   }

   function chi2Q( $x,  $v) {
      $m = (double)$x / 2.0;
      $s = exp(-$m);
      $t = $s;

      for($i=1; $i < ($v/2);$i++) {
         $t *= $m/$i;
         $s += $t;
      }
      return ( $s < 1.0) ? $s : 1.0;
   }
}
```