Feature Extraction for Spam Classification

by

Michael Davy

A dissertation submitted to the University of Dublin
in partial fulfilment of the requirements for the degree of

Masters of Science in Computer Science

Department of Computer Science,
University of Dublin, Trinity College

September, 2004

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signature of Author:

_____

Michael Davy.

Wednesday, 08 September

2004

Permission to Lend and/or Copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signature of Author:

_____

Michael Davy.

Wednesday, 08 September

2004

Acknowledgements

I would like to thank my supervisor, Pádraig Cunningham for his help and guidance throughout. I would also like to thank Sarah Jane Delaney and Gavin O'Gorman for their help in developing and testing the system. Apologies to my friends who must think I have disappeared off the face of the planet. Finally I would like to thank my fellow students who have made the whole year an enjoyable and memorable experience.

Abstract

E-mail has emerged as one of the primary means of communication used in the world today. Its rapid adoption has left it ripe for misuse and abuse. This came in the guise of Unsolicited Commercial E-mail (UCE) or as it is otherwise known Spam.

For a time spam was considered only a nuisance but due mainly to the copious amounts of spam being sent it has progressed from being a nuisance to become a major problem. The volume of spam has reached epidemic proportions with estimates of up to 80% of all e-mail sent actually being spam.

Spam filtering offers a way to curb the problem. Identifying and removal of spam from the e-mail delivery system allows end-users to regain a useful means of communication. A lot of research in spam filtering has been centred on more sophistication in the classifiers used. This thesis begins to investigate the impact of applying more sophistication to lower layers in the filtering process, namely extracting information from e-mail.

Several types of obfuscation are discussed which are becoming ever more present in spam in order to try confuse and circumvent the current filtering processes. The results obtained by removing certain types of obfuscation show to improve the classification process.

The main theory under investigation was the impact of pair tokens on the classification process. It is quite reasonable to think that pairs of tokens will offer more value than single tokens alone. For example “enlarge your” seems to suggest more information than single tokens alone. Results obtained show conclusively that pair tokens offer no value and in fact increase error over three independent data sets.

Table of Contents

Table of Figures

List of Tables

List of Equations

1 Introduction

1.1 Background Information

The cornerstone of modern life is communication; from morning to evening we expect to be able to communicate instantly with others. E-mail has progressed rapidly from its original beginnings as a communication medium for academics to its current role as one of the most widely used methods of communication throughout the world. The notion of sending a conventional letter to a person in another country seems like something from the dark ages. Practically everyone who uses the internet seems to have an e-mail account. Business has leveraged this technology as a communication channel for its employees (it's rare to find any company which does not have an internal e-mail system).

E-mail account are given away free and can be checked from anywhere in the world at any time. Its popularity is due to the fact that it's fast, cheap and reliable. The e-mail delivery system was designed to be open, meaning that anyone could connect to a server and start sending e-mail.

With such a wide user base it was only a matter of time this resource was exploited. As can be seen in other forms of communication, when a return on investment is attainable, advertisements appear; television commercials, radio announcements, news paper advertisements (some newspapers seem to consist of 90% advertisements and 10% content).

A new menace was released into the world, one that would start as a nuisance but would soon grow to be a major problem. Unsolicited Commercial E-mail (UCE) or as it's more commonly known "Spam". Personally I define spam as 'Unsolicited or unwanted commercial e-mail'. Unsolicited is important, if you never asked for it then I consider it spam,

alternatively you “opt-in” to receive the e-mail in which case I do not consider it spam.

Spam tends not to originate from large businesses; rather it normally comes from very small businesses and from individuals who are trying to make some money from gullible people. The reason these people choose spam as their advertisement medium is because it’s so cheap. It costs practically nothing to send spam compared with an advertisements campaign on television or a newspaper. Because it’s so cheap, they need a miniscule return rate, for example, if 1,000,000 spam is sent out, if 5 people respond and buy the product then the campaign costs are covered and the business could be in profit!

Large or more established companies typically do not send spam as it would damage their reputation. They tend to stick to the more mainstream communication channels such as radio, billboards, television etc…

Spam typically advertises small, easily transferable goods and services such as Viagra or Software. There are other flavours of spam including: scam spam and pornography spam. The later advertises membership to a website offering pornographic images. The former is a little bit more devious.

The best known scam spam are 419’ers, scam spam is where the spam sender (a spammer) tries to trick gullible individuals out of money by promising them more money in return for their ‘initial investment’. This type of spam allegedly originated from a group of Nigerian spammers who send out this type of spam to millions of e-mail accounts. The revenue expected from this type of scam in 2003 was $2,000,000,000 [ePrivacy03].

The problem that has emerged is not that spam exists or that gullible individuals are being ripped-off, rather it is the volume at which spam is

being sent. Figures of 80% of all e-mail which is actually spam, taking MSN™ as an example, they block 2,400,000,000 spam messages a day! Companies have to purchase additional servers and storage to cope with the pressure on their e-mail systems. A figure of $25.5BN is expected to be spent by companies around the world in purchasing this additional equipment. All figures quoted were obtained from [ePrivacy03].

1.2 Hypothesis under Investigation

This thesis investigates the spam filtering process. A lot of current research is centred around the classification stage developing more elaborate and advanced classifiers but little work has been done in the stages that lead up to the classification process. These stages extract the information from the e-mail or spam ready for use in the classification process. “A worker is only as good as their tools” and in the same way a classifier is only as good as the features it works with. Better features extracted should result in a more accurate classification.

This thesis investigates the new methods used by spammers in order to circumvent modern spam filters by disrupting the tokenising and classification processes. The removal of obfuscation and inclusion of other non-token features may help to increase classification accuracy.

The use of pair tokens should offer more information to the classifier. For example the feature “click here” offers more information than “click” or “here” individually.

1.3 Document Roadmap

The rest of this chapter presents an overview of the remainder of the thesis.

In chapter two spam sending techniques are discussed with particular attention paid to the types of obfuscation used to disguise spam and web bugs used to track the success of a campaign.

In chapter three spam filtering techniques are discussed focusing on the current popular methods of filtering spam including rule based and content based filtering.

In chapter four, the apparatus is presented. Each stage in the process is examined and explained. Methods in the system for dealing with obfuscation are discussed.

In chapter five, the experiments conducted and results obtained are discussed. The results are displayed diagrammatically and a short discussion on what was noted given.

In chapter six, conclusions are drawn on the results obtained from the experiments. Discussion is given on other related work in the spam arena. Finally future work is presented.

The reader is directed to the glossary in order to familiarise themselves with the names and phrases used commonly when discussing spam.

2 Spamming Techniques

It is becoming ever easier to start spamming. Websites offer clients all the necessary tools in order to send “Bulk e-mail”. Not only this, they also offer lists containing millions of valid e-mail addresses.

If worried about legal reprimands, the spam sender (spammer) can conceal their identity easily with the use of open relays, open proxies and even zombies. The later is a new trend where spammers team up with virus writers so that infected machines act as spam relays – all unbeknownst to the owner of the machine.

Spam itself has mutated in order to have the same impact it did in previous years. This mutation was caused by the introduction of filtering systems which separate spam from legitimate e-mail and delete it. It never reaches the end-user thus the return rate on the spam campaign may drop.

The mutations were made in order to try to circumvent spam filters. The inventive ways in which spammers have mutated spam is impressive. This leads the author to think that there is still a large incentive to spam – why bother spend time and resources if it’s all futile.

Below a description of some of the mutation techniques noted while investigating spam.

2.1 Beating the filters

Spammers have to continually try to circumvent filters. In the past when the types of filters used were simple rule based systems it was often quite easy to circumvent these. This was due to the fact that with rule based systems constant updating of rules was required. Often this task fell to

administrators and end-users; they simply did not have the time or the expertise to construct effective new rules.

A common rule was that if an e-mail contained a certain word then it was considered a spam. Spammers simply replaced such words with a synonym or misspelled the word. The latter is not as disastrous as one might think. Readers generally can still understand a piece of text even if spelling mistakes are strewn across the text [Rawlinson76]. “stlil esay to uednrstand”

With the advent of more sophisticated filtering techniques such as Bayesian based filtering the changes the spammer needed to make became more sophisticated.

2.1.1 Obfuscations

In order to try and disguise certain words and phrases that may be used to easily distinguish spam from legitimate e-mail, spammers have resorted to obfuscating these words in order to try and confuse filters. Some of the more prominent methods of obfuscation are described below.

2.1.1.1 Added Punctuation

A common type of obfuscation is where punctuation is used in between every letter of a word. For example the word Viagra is obfuscated to V-i-a-g-r-a. For the human reader this is still perfectly legible but for classifiers this obfuscated version of the word is considered a completely different feature. The feature may not be recognised as a strong indication that the e-mail is spam.

With Bayesian classifiers this is a particular problem. Not only does it cause error rates to increase but it also means an increase in the space required to store all previously seen features. It may be argued that these types of features are more descriptive of spam as they occur more commonly in

spam. If a generic identification of this form of obfuscation is available then this should help increase filtering accuracy without the need to store all obfuscated versions of a particular word.

2.1.1.2 Misspelling

Features such as “porn” are becoming very strong indicators of spam. This means that the accuracy of filters using these features is becoming ever more accurate. In an effort to combat this and circumvent the filter, spammers have resorted to misspelling words.

Surprisingly the legibility of the spam remains high! A study [Rawlinson76] has shown that misspelled words do not dramatically impact on the legibility of a given text. Spammers also tend to substitute letters with similar looking numbers (for example the letter “I” is often replaced with the number 1). Taking the feature “porn” some common misspellings of it include “p0rn”, “pron” and “pr0n”.

Misspelled words have a similar affect to classifiers as added punctuation obfuscated words have. The number of possible features increases and the accuracy of the classification decreases.

Again there is an argument to keep these misspelled words as features as they are more descriptive of spam in general. Many people do not use correct and complete English when constructing an e-mail. Most do not even use a spellchecker. These subtle spelling mistakes may provide to be quite useful. Note this is in contrast to spammers deliberately trying to obfuscate words by replacing letters with similar looking digits or punctuation.

If there was a way to spot the use of such obfuscation methods then this may prove useful. With the use of regular expressions or examining the

constituent characters of a word it should be possible to detect misspellings such as “p0rn” (note the use of zero rather than the letter o).

2.1.1.3 Effectiveness of Obfuscation

The introduction of obfuscation could actually help to increase accuracy of filters. The presence of obfuscation could be used as a feature when determining if an e-mail is legitimate or spam.

2.1.2 Salad

With the advent of more sophisticated filtering techniques such as Machine Learning and in particular Bayesian and hash based filtering, spammers needed to change their spam in order for it to circumvent the filters. They began inserting random text into their spam in order to circumvent these classifiers. This random text is commonly known as salad.

The effect that salad has on a spam is that it will introduce a number of words which should confuse the classification process with the result that spam is not as distinctive from legitimate e-mail. It’s should be harder for the classifier to distinguish legitimate e-mail from spam resulting in diminished accuracy of the classifier. In addition there is extra storage requirement for some classifiers as they will need to store the number of occurrences of these salad words.

The salad the spammer introduces will typically be selected very carefully. It can not be words or phrases typically found in spam otherwise this would not make spam less distinctive. They generally used either words from a dictionary, quotations from novels, quotations or indeed content from a website.

The result should be that the spam begins to resemble legitimate e-mail when considered by the classifier. Below a discussion is given on the two

types of salad that seem to be popular in spam in corpora collected for these experiments.

2.1.2.1 Random Character Salad

This kind of salad is just a sequence of random characters. The sequences are meaningless but are considered as words by classifiers. The random characters introduce randomness into each spam. The characters are often found at the end of the subject line and are normally hidden in the message body as white space (white text on white background) or as HTML tags. This type of countermeasure is normally directed towards hash based classifiers but may be effective against some Bayesian classifiers.

T<kmrkcygdiboudjb>he o<kexxwqdewrdhfbs>n<ktcgsdocqmwjir>ly<kmxdwjobcxhjh> so<kqsaqfwtfaqdw>lut<kvnoibncqsltsvd>ion
Random Character Tag Salad

ixrr f aoamy zaqr wolfgang cutesy discus astor folksy frsnbpoybvcacgaarpnfb yqprnbhijjpvmco qhtl buchanan dividend dybuilding
Random Character Salad (Hidden text)

Figure 1 - Random Character Salad

Hash based classifiers classify a new case based upon whether a hash of the new instance matches any know hash of a spam. If so, the new case is assigned the classification of spam. Hashes used are generally cryptographic hashes such as SHA-1. The Achilles-heal of this genre of classifier is that the hash algorithms obey the strict avalanche effect [Feistel73] [WebsTav85] which states that a single bit change in input will cause at least half of the output digest bits to change.

Hashes of known spam could be distributed to multiple peers creating a collective of known spam. The task of classification then becomes quite

simple. There is little chance of false positives since each unique input text will hash to a unique and distinct output value (i.e. multiple copies of the same spam will has to the same value).

The problem is that in order to circumvent these filters, the spammer needs to only make subtle changes in the spam. The introduction of randomness in each spam effectively means that each spam will hash to a different value. Even though a collection of spam will look practically identical they will differ in their random salad values and will thus each have different hash values. The effectiveness of the hash is reduced dramatically. Any spam with randomness within it will bypass the filter. Further work on this area has been to develop a fuzzy hash algorithm. This type of algorithm would not display the strict avalanche effect, thus would not be susceptible to the above problems.

Random character salad is used in order to try to bypass Bayesian classifiers. The addition of the random characters should help to neutralise the probability that the e-mail will be classified as spam. This is due to the fact that this salad may contain features that occur often in the user's legitimate e-mail. The spammer would have to guess a feature that occurs in legitimate e-mail and doing this for millions of users may not be feasible [Cumming04].

2.1.2.2 Random Word Salad

Another type of salad that is becoming more prevalent in spam is the addition of random words at the end of the message body or in HTML comments. In the early adoption of this technique spammers would just use random words but it is becoming more common for the addition of portions of text, such as excerpts from novels or even quotations.

B<flyer>ecau<weapon>s<tub>e you can add mor<maritime>e <belate>to your l<almaden>if<serpent>e.
Random Word Tag Salad

hiphopnbenzspeak egotist deferring combustion scandal rhombohedral seder ucla altogether amperage debonair dionysian politic frustrate checkbook arizona cadaverous
Random Word Salad (found at bottom of spam)

Figure 2 - Random Word Salad

The affect of this type of salad is the same as the affect random characters has on hash based classifiers. With the introduction of randomness to the spam, the effectiveness of the hash to classify future spam will diminish rapidly. As mentioned above the use of a fuzzy hash algorithm may help in combating salad.

Bayesian classification relies on dissimilarity between features found in spam and features found in legitimate e-mail. With the introduction of random words, chosen from a source that is itself not related to the spam, it may affect the accuracy of the Bayesian classifier. If a spammer can correctly guess words which are indicative of legitimate e-mail then this may alter the classification process. Guessing the correct features to affect a large population may not be feasible [Cumming04].

2.1.2.3 Effectiveness of Salad

Salad can be used to circumvent hash based filtering techniques but whether or not it can be used to defeat Bayesian based filters is another question. In regards to hash based filtering random salad will randomise the hash value for each spam. This means that each spam will have a different hash value, negating the benefits of hash based filtering.

For Bayesian based filtering it may cause no decrease in filter performance. The only way that random salad would impact on Bayesian type filters is if the random salad were to contain legitimate features. With Bayesian filters, a list of spam features and a list of legitimate e-mail features are maintained. The spammer would need to append sufficient amounts of your individual legitimate e-mail features.

In order to get this to work they would need to either obtain your list of legitimate e-mail features or train their spam against your Bayesian filter. Dr John Graham-Cummings gives a good explanation of the process in a talk he gave to the Spam Conference [Cumming04]. Salad should lower the probability of the spam being classified as spam but this seems not to be the case.

2.1.3 Web Bugs

A web bugs (also known as: web beacons, pixel tag or clear gif) is an image which is normally very small, that is placed in and e-mail or on a website and is used to monitor the behaviour of the recipient of the e-mail or the visitors to a website.

In its most simple form, a web bug is a one pixel by one pixel image. Users rarely notice its existence. When browsing a website the image is normally downloaded when browsing a HTML page. Within the source of the web page there is an img tag calling the web bug, for example.

```
<img src="http://mydomain.tld/images/myBug.gif">
```

Figure 3 - Simplistic Web Bug

To the casual observer this may seem to give no information at all, but combined with the fact that a HTTP server will log all activity to the website

it becomes a valuable tool. When your browser loads the web bug it must contact the server and download the image. The HTTP server will log your IP address and may also log other information such as the browser used and the time you downloaded the image.

With e-mail, web bugs are more intrusive. They work in e-mail because of the growing trend of HTML styled e-mail. Rather than sending plain text e-mails, many people opt to send HTML styled e-mail because they offer more formatting power, allowing italics, bold, and coloured fonts. The most popular MUAs all allow for HTML styled e-mail and in addition to sending styled e-mail they are able to parse and display HTML styled e-mail. It is for this reason that web bugs work in e-mail. Not only does the MUA display the HTML, it does it automatically. Preview panes automatically render the HTML, activating the web bug without you even opening the e-mail.

With ordinary web bugs placed in a spam, spammers can detect what proportion of their spam has been opened and from what IP. This is useful since they can determine what percentage of their spam has reached its intended audience. If a spammer has sent out 2, 000,000 spam and there are 1,000,000 downloads of the web bug then they know that half of their spam list are active, or that only half of their spam have reached their destination.

A more sophisticated form of web bugs has begun to appear in spam. The spammer can not only gather passive intelligence such as your email reading habits, IP address, operating system, and browser, but can actually verify your e-mail address. Instead of a URL pointing to some static content such as a gif or jpeg, the URL points to some dynamic content. Typically the server runs a CGI, which will take some predefined parameters. HTTP allows for information to be sent via the GET and POST commands, spammers have hard coded into each spam a web bug which has as its argument the e-mail address to which the spam was sent.

For example the following web bug contained in a spam sent to joe@bloggs.com. The web bug contained within the spam has as its argument the e-mail address joe@bloggs.com. When the MUA displays the e-mail it will load the web bug from mydomain.tld where a program running on the web server will validate that joe@bloggs.com is a valid e-mail address and has opened the spam.

A	<img src="http://mydomain.tld/haverst.pl?email=joe@bloggs.com">

B	<img src="http://mydomain.tld/haverst.pl?email=3dfa21d76b3c4d">

Figure 4 - Advanced Web Bugs

Web bugs such as A are somewhat easy to spot, so spammers are now trying to conceal this by using a hash of your e-mail instead, shown in B.

The spammer now knows that the e-mail account is active and that the spam is successfully being delivered (i.e. no spam filtering taking place, or if there is, then the spam has circumvented it). They can then continue to send spam to this address and sell the address on to other spammers. This verified accounts cost more than unverified addresses.

2.1.3.1 Web Bugs in the Work Place

An interesting aside to the use of web bugs is in their use within an organisation. Administrators can spam their own mail-server with a specially prepared message containing web bugs. Since the web bug does not need to be on the same server as the one delivering the content, they could place the web bug on their own corporate web server. This could be used to track which employees are viewing unsuitable material during office hours. A suitably enticing subject may be enough to catch an unproductive employee (e.g. “Play games which behind your bosses back” or something along those lines).

3 Spam Filtering

There are many ways to deal with spam. Filtering is a highly popular technique. It involves selecting and removing spam from legitimate e-mail. Discussed below are some of the filtering techniques.

3.1 Human Filtering

For most people the only spam filtering is done by hand. When presented with an inbox crammed full of spam, the general reaction is to select all the spam and delete it. A human is able to determine in a fraction of a second whether or not an e-mail is spam or legitimate.

This technique has the advantage of high accuracy, with a low rate of false positives but it has some serious problems. Human filtering is time consuming and extremely tedious. Most end-users do not want to wade through their inbox looking for spam and deleting it by hand.

In addition this scheme is not 100% effective. Users may accidentally delete a legitimate e-mail when purging the inbox of spam. This is known as a spam-spasm. With the overwhelming amount of spam compared to legitimate e-mail it's not uncommon for users to accidentally select legitimate e-mail when selecting groups of spam for deletion.

From my own personal experience, the number of spam received per day is sometimes as much as 50:1 (50 spam messages for every one legitimate message). The trend is for this ratio to rise. The ease at which spam can be sent means that more spammers are starting up [Judge03]. A conservative estimate would state that half of all e-mail sent per day is in fact spam!

With the growing popularity of spamming, automated methods are required to regain a useful e-mail system.

3.2 Rule Based Spam Filtering

The first automated filtering techniques to be discussed are those which use a set of rules to classify e-mail as spam or legitimate e-mail. These techniques used to be in vogue but have of late been superseded by the techniques discussed later in this chapter.

The rule based filtering techniques can be applied at either the MTA level or the MUA level. First discussed will be the MUA level filtering techniques then discuss the MTA level filters.

3.2.1 MUA Rule Based Filters

Most e-mail clients (MUA) have some sort of feature for categorising e-mail based on a set of rules determined by the user. These rules can be constructed to examine an e-mail message's body for keywords or phrases given by the end-user. A common use of such rules is to categorise newly arrived e-mail into a specific folder. For example, some users have a folder for work e-mails. A rule could be setup to transfer newly arrived e-mail that contain the word "job" into the work folder.

The same technique can be applied to categorising spam from legitimate e-mail messages. The user could create a folder called spam and define a number rules that would transfer a newly arrived e-mail to the spam folder if it triggered them. Such rules could look for specific words in the content of the e-mail, or look for punctuation being used in the subject of the e-mail, or note the content type of the e-mail.

While this technique does work well, it does have a serious problem. The rule set needs constant updating and refinement. This is because most spammers know that this kind of filtering is popular, so in order to circumvent the filtering they use obfuscation techniques. Some common obfuscation used is misspelling words. For example if you had a rule which

said “Any e-mail with the word “Free” in the subject line, put in the spam folder”. Obfuscating the word “Free” may result in “F*R*E*E” which will bypass this rule and will end up in the user’s inbox.

Not only are spammers circumventing the rules by obfuscation, legitimate e-mail could also be incorrectly classified because of a rule with is too restrictive. For example “Any e-mail which does not have an Attachment should be put in spam folder”. Clearly this is too general and will cause a lot of false positive results.

3.2.1.1 Whitelist

Another MUA level rule based filtering technique is that of whitelists. A whitelist is a list containing a collection of contacts from which you will accept e-mail messages from. If an e-mail arrives but does not come from one of the contacts in the whitelist then it is rejected (placed in spam folder). While this technique is effective for some users it is clear to see it has faults.

Any mail sent by a stranger will simply be incorrectly classified as a spam – in other words it’s a false positive! In all but a few scenarios it’s inconceivable to know *a priori* all contacts that will send you an e-mail.

There a slight variation on this scheme that incorporates a challenge response mechanism to allow users to be added to a user’s whitelist. The basic scenario is that when a stranger sends a message to you, the MTA/MUA will automatically respond with a challenge and until such time when the stranger responds with the correct answer to the challenge, the e-mail is not delivered. This challenge is normally just a request for the stranger to reply to the challenge (The challenge contains some redundant and random information which will not allow spammers to circumvent it).

Again there are noticeable flaws in using this technique. Firstly the MTA/MUA needs to have new software incorporated to it – a daunting challenge itself with the scale of the internet. Next the extra processing involved in sending a challenge for every e-mail received by a stranger is not negligible. Finally, users simply may not respond to the challenge! Either by lack of motivation or by being uneducated (not everyone who uses e-mail is a computer scientist). These count as false positives and reduce the effectiveness of the communication channel.

One technique that was starting to become more evident at the time of writing was spammers forging sender information to be from the same domain as the intended recipient. Most whitelist tend to contain a lot of addresses from other people on the same domain. As an example, suppose a spammer wanted to spam joe@bar.com they would simply forge the sender information to show that the mail came from mary@bar.com. This may be enough to circumvent the whitelist. The chances that users on the same domain communicate with each other is much higher, thus the chances of there being an entry in the whitelist becomes higher.

3.2.1.2 Blacklist

The antipode of the whitelist, a blacklist contains lists of known spammers. Essentially when you get a spam, you add the sender of the spam to the blacklist. The entire domain of the sender of the spam can be added to the blacklist. Newly arrived e-mails are checked, if their sender is on the blacklist the e-mail is automatically classified as spam.

As with whitelist there are flaws with blacklists also. The major problem stems from the fact that spammers tend to forge header information in their spam. The sender information is generally forged, meaning that perhaps innocent people are added to a blacklist but more importantly the effect the blacklist will have is diminished dramatically. Since the spammer

can easily forge the sender information they can circumvent the blacklist with ease.

3.2.2 MTA Rule Based Filtering

Some of the techniques described for MUA rule based filtering can be applied at the MTA level. Filtering at this level can achieve some economies of scale but also contributes some problems. Since spam by its nature is sent in bulk, blocking the sender can dramatically reduce the number of spam needed to be stored and delivered.

3.2.2.1 Distributed Blacklist

Blacklist at the MUA level only work for a single user. If applied at the MTA level a single blacklist could work for all users in a domain or sub domain. This would reduce the burden on each user maintaining their own blacklist.

Real-time black lists (block lists) commonly known as RBLs are emerging as a strong anti-spam technology. The idea is simple; a central repository keeps track of Internet Service Providers which are known to have spam activity on their networks. In other words, a list of ISPs' who are known to allow spammers use their network to connect to the internet and send spam are listed in a repository.

An MTA can be configured to consult this repository when a new e-mail has been presented for delivery. When a host connect and tries to get an e-mail delivered its IP (Internet Protocol) is recorded and checked to see if it is in the repository. If the host's IP is listed in the repository then the delivery of e-mail is refused, otherwise deliver proceeds as normal.

The central repository is maintained constantly and the IP of ISPs' may be added or removed if they are found to have harboured spammers or removed spammers from their network.

The principle behind this scheme is that ISPs that are known to harbour spammers are punished, while ISPs who do not encourage spamming are left unaffected. This scheme aims at penalising ISPs for allowing spammers to use their networks. This should force the ISP to increase security and not encourage spamming on its network. In addition it only affects mail delivery, other internet services are unaffected.

This is a good approach but it does have some problems. The first problem is that the central repository must be maintained and it must be maintained by an unbiased organisation. One could imagine a scenario where a rival could persuade the maintainer to add its competitor to the block list. The competitor's reputation would be damaged and it could loose business.

A common scheme is where system administrators will report the IP of a host it has received spam from. The maintainer should verify the fact that the IP in question did indeed send the spam and then add it to the block list. There are a couple of suppliers of RBL in the market today and the 'aggressiveness' of the maintainer to add an IP to the list is different between each provider (some will add an IP address quicker than others).

The major problem with RBL and related technologies is that it punishes legitimate users as well as spammers. Blocking all mail from an ISP is draconian and heavy handed solution. Only a small minority of the users of the ISP have caused the problem yet every user of the ISP is punished!

Other types of RBL have emerged where they are more selective on what is stored in the central repository. Some store IP addresses of open proxies, which spammers use to conceal their identities when sending spam. Others

store the dynamic IP addresses typically assigned to dial-up users in an effort to reject mail from unusual sources. These are a step in the right direction in the author's opinion.

3.3 Content Based Spam Filtering

Spam will typically have distinctive content, which should be easy to distinguish from legitimate e-mail. In the previous ontology of spam filtering techniques, only parts of the content were examined, if at all. In the case of real-time block lists the e-mail may never be examined.

Categorising e-mail based on its content seems like a logical progression from simplistic rule based approaches. This would help reduce error rates as legitimate e-mail would not be blocked even if the ISP from which it originated is on a real-time block list. In addition the presence of a single word (such as Viagra) should not alone cause the e-mail to be classified as spam.

With the aid of Machine Learning techniques, rule sets would only need occasional refinement and in most cases the refinement is automated, meaning less hassle for end-users (most non computer science end-users typically do not care about how the spam filter works). As the spam changes over time, the machine learning techniques could help track the changes, automatically adjusting itself to increase accuracy against the new variant of spam.

Machine Learning has brought a lot of new techniques to the problem of spam classification. A full taxonomy of all the techniques and their theoretical background is beyond the scope of this paper. Discussed below are the most prominent techniques that are currently making a serious impact on spam classification.

3.3.1 Content Hash Based Filtering

It soon became apparent that the content of spam was quite similar. In addition the same message is sent in bulk, often in groups of one million spam e-mails at a time. Being able to quickly identify spam using a short hand notation would be a succinct advantage. The ideal choice of this short-hand notation was a hash (also known as a message digest).

A hash of the content of the spam was computed, resulting in typically a 128 bit (or 160bit) representation of the spam. When a new e-mail arrives its hash can be compared to a list of know spam hashes. If there is a match then the e-mail can be deleted without fear.

The true strength of this technique only becomes apparent when we introduce a distribution mechanism for known spam hashes. A similar concept to the RBL approach, distributing known spam hash values allows a MTA or MUA to delete a spam without it ever having received it before.

Anti-Spam has often been described as an arms race with spammers. As new techniques are developed to combat spam, spammers will try to circumvent these. In this case the spammers attacked the short-hand notation. A hash algorithm such as SHA-1 obeys the strict avalanche effect. This states that for every 1 bit of change in input at least half the bits in the digest should be affected.

The strength underpinning the distributed hash technique was that spam content was the same. In order to circumvent the hash based filters spammers started to introduce some random variables into spam. Typically this would be random characters placed either in the body or on the subject line. The result was that each and every spam now had a unique hash value. When the hash of a spam was computed, its hash would not match that of any known spam since it contains some random variables. Due to the strict avalanche affect, the output was completely different for each

spam which has some random content. Efforts are underway to make fussy hash algorithms which would not be as prone to random variables in the content of the spam.

3.3.2 Bayesian Filtering

Examining spam and legitimate e-mail it becomes clear that the style and language used is quite different. Words that occur frequently in spam do not occur frequently in legitimate e-mail. This fact allows us to use some statistical tools to help classify spam from legitimate e-mail.

An early, influential paper on the topic was [Graham02]. In his paper Graham described an algorithm for using Bayesian statistics (used in a very much simplified manner) to identify spam with an extremely high accuracy. The idea was to construct a probability for each word, on whether it was indicative of a spam.

A high probability indicated that this word was normally present in spam but not present in legitimate e-mail and vice versa. Combining the probabilities for each word in a given e-mail it was possible to calculate the probability for an entire e-mail on whether it was a spam. A threshold could be set, so that any e-mail with a spam probability lower than this threshold would not be classified as spam. Any e-mails which exceeded this threshold would be marked as spam.

The Bayesian probabilities of each word being indicative of spam were based on corpora of legitimate and spam e-mail being stored by the user. By examining both the size of the corpora and frequency of each word within the two corpora the Bayesian probabilities could be calculated. Bayesian statistics allow us to infer probabilities about future events by looking at past events (The probability that X will happen given that Y has happened). This exactly matches what we are trying to do in spam

classification. Based on the information contained in the corpora, we will try to infer a probability about whether a new instance of e-mail presented to the system is a spam or a legitimate e-mail.

In the arsenal of weapons against spammers, Bayesian statistics based filters are one of the 'big guns'. Since their introduction, Bayesian filters have proven to be highly effective in filtering spam, offering impressive accuracy of up to 99.999% - rule based filters do not offer such a high accuracy rate.

In any arms race, if your competitor has developed a devastating new weapon, you must react to either develop a better weapon or find a way to reduce the effectiveness of their weapon. Spammers had to invest some time and effort in order to try and circumvent Bayesian filters.

The problem spammers faces was that their spam contained words which were very indicative of spam, thus carried a high probability. When the calculation of the probabilities of all the words in a given e-mail was done, spam contained a lot of high probability words, thus increasing the probability being calculated for the entire e-mail.

Spammers began to realise that their spam was not reaching its intended target because of Bayesian filters. Some began to include very neutral words in their spam in order to lower the overall probability. Words such as "hello" occur frequently in both legitimate and spam e-mail. They also began to obfuscate the words in their spam which had high probabilities, such as "Viagra" was obfuscated to something like "V-i-a-g-r-a".

The goal was to lower the probability of their spam to such an extent that it is lower than the threshold, thus will not be classified as spam and allowed to the user's inbox. This trend is happening in more types of spam and with different variations of methods to lower the spam probability of the

message. Some have resorted to adding in some quotes by famous authors at the end of each spam.

3.3.3 Instance Based Filtering

Instance based learning is quite different to the other types of machine learning. It does not construct a model of some target function from the examples provided, rather it stores the examples and only generalises when a new instance is presented to the system for classification. The instance is compared against all the stored examples to find its similarity with them. From this metric a classification can be assigned.

K-Nearest Neighbour is one of the most used and theoretically easy to comprehend, instance based machine learning. Training data presented to the system is stored. When a new instance is presented, its similarity to each of the training data instances is calculated. The K training instances which are nearest to the new instance are selected and can participate in classifying the new instance. A commonly used approach is that the classification that occurs most commonly in the K nearest neighbours is assigned to the new instance (The mode of the classifications in the K nearest neighbours).

Instance based learning is called a lazy learner since it does not process the training data until a new instance has to be classified. This can mean a significant overhead at runtime, but has the advantage of a dynamic target function – it changes for each new instance presented for classification.

Its use in anti-spam technology is not widespread due to the fact that instances need to be stored. This can lead to a large overhead in storage. In addition the lazy approach to generalising means that most processing is done at classification time (on-line). When dealing with a large volume of e-mail, this processing overhead can be too high. An eager learner which

does most of its processing at training time (off-line) can be much more efficient when dealing with large volume of e-mail.

3.4 Impact of Anti-Spam

The fact that spammers are reacting to filtering is proof that it is making an impact. The time and effort spammers need to invest is greater now that it was in the past. This will increase the costs associated with sending spam and will hopefully reduce the problem of spam.

3.4.1 A World Without Spam?

Anti-spam sole objective is to stop spam, or at least make it manageable. Spammers' sole objective is to deliver messages to as many people as possible for as little money as possible. I do fear that as one avenue is made more difficult for spammers, they will choose the path of least resistance and move to another communication medium in order to send their message.

We have already begun to see spammers teaming up with virus creators. Computers are infected with a virus and then become an open relay for spammers. These are called zombies. The owners of the computer typically do not know their computer has been compromised. Spammers can command the computer remotely to begin sending spam.

This is a worrying trend and shows that spammers will go to extraordinary lengths in order to save money and still deliver their marketing message.

4 Apparatus

4.1 System

The system is composed of several different subsystems each of which has a specific role or function. Raw data is input to the system in the form of Eudora MBX files. These are processed and each e-mail is stored in an internal object representation.

Once the e-mails have been stored in an internal representation they can then modified by the text processing subsystem. Its role is to remove obfuscation and construct non-token features. The processed e-mails can then be tokenised, which is done in the tokenise subsystem. Its role is to break up an e-mail into its features and count occurrences. The tokenised e-mail information is stored in a new data structure which can be used by the next subsystem.

Classification is the last subsystem; its role is to assign classification to an e-mail being presented to it. The classifiers will assign classifications to new instances based on generalisations from their training data. Two forms of classifier have been implemented; one being an eager learner (Bayesian) while the other is a lazy learner (K-Nearest Neighbour).

Surrounding and controlling the ensemble of subsystems is a validation system. Its role is to conduct the experiments and collect results. A K-fold cross validation system was implemented. It will conduct the experiment K times then collect the results from each run. Finally it will calculate the arithmetic mean of the results. In the system the validation measures the error rate of the classifier. It also collects the false positive and false negative rates. Below a detailed description of some of the subsystems in the system is given.

4.2 Mail Input

4.2.1 Corpora

In order to conduct the experiments a collection of both spam and legitimate e-mail must be gathered. These corpora are used both in training and in testing the proposed classification techniques. A number of collections were used; this was in order to try to prevent local anomalies overfitting the classification techniques.

Each data set consisted of a corpus of spam and a corpus of legitimate e-mail. The tests were run of three data sets. The collection consists of archived e-mail from at least two distinct end-users.

4.2.2 Eudora MBX files

Most users tend to use Microsoft Outlook (or Microsoft Outlook Express) but this presented a problem. A proprietary file format is used by both. It is not an open standard thus it is impractical to collect mailbox files from users who used these MUAs.

There are projects underway on sourceforge.net to make a conversion tool in order to allow users to switch from Microsoft Outlook/Outlook Express to another MUA allowing them to import all their old messages. LibPst is a part of the [ol2mbox] project and is a tool that will convert Microsoft Outlook PST files into maildir UNIX standard. Using LibPst it was possible to convert from PST files to Eudora MBX file format.

The Eudora MUA by Qualcomm software is a client which uses the standard mbox file format when storing e-mail messages on the user's hard disk. It is quite straightforward to read and parse this file format as Eudora does much of the parsing itself. Attachments are automatically stripped from the

e-mail and stored on disk. This means less time would need to be spent developing complex MIME parsing classes.

In addition to the above advantages Eudora allows a user to import old messages from a Microsoft Outlook PST file. This would allow me to convert e-mails stored in PST files to the much more manageable Eudora MBX file format (mbox format). However this process was never foolproof and often fraught with complications and errors. Encrypted PST folders could not be read by Eudora and sometimes the import feature simply did not work. Manual manipulation was often needed for satisfactory construction of MBX files.

4.2.3 Spam Collection

In order to be as accurate as possible, wild spam was collected for use in experiments. Wild spam is spam collected directly by myself at a number of e-mail accounts. This is in contrast to second hand spam which is defined as spam contributed by other people (they forward or bounce the spam to you). An example of second hand spam is that of the spam corpus in the spam archive.

The reason for using wild spam in preference to second-hand spam is due to the fact that the header information may be lost. In addition the act of forwarding the spam may introduce some anomalies that could affect the results obtained. In addition wild spam requires less parsing to remove redundant information.

In addition the use of recently collected spam was used in each of the three data sets. Newer spam is harder to classify due to the obfuscation techniques employed by spammers. Using only new spam should be more of a challenge than using older – possibly easier to identify – spam.

4.3 Text Processing

Noise is defined as being obfuscated and malformed material present in spam. This was a cause for concern when designing the system. It may seriously affect the tokenisation and the performance of the classifiers. A method of removing as much noise from the corpus before the classification stage would be advantageous.

Regular expressions offer a lot of power when dealing with certain types of noise. Throughout the system regular expression have been used to identify, remove and replace noise in the data before it is tokenised. This should help to improve tokenisation and the performance of the classification since better features are being considered.

4.3.1 HTML Removal

One type of noise where this power (regular expressions) can be leveraged is dealing with HTML comment salad. A simplistic regular expression could be constructed in order to identify all HTML comments. Since HTML comments are not displayed to the end-user its complete removal should not cause any damage.

Within the system a text removal method was developed. It took a supplied regular expression which it used to search for matches. Once a match is found, the match is removed from the text. The output of the method is the input text with all the matches of the regular expression removed. In the case of HTML comment salad this would identify all HTML comments and remove them completely from the e-mail.

4.3.2 HTML Replacement

The vast majority of spam is HTML (MIME type text/html). Within the HTML there is a lot of noise in the form of comment salad, tag salad, word

obfuscation, random word and random character salad. Below is an example of each:

<B>Che<!-- benthic -->ap V<!-- bounty -->ia<!-- coffey -->g<!-- physik -->ra<FONT COLOR="#FF0000">
HTML comment Salad

<CENTER><P><FONT size=+0><B>GET<!k> Y<!r>OU<!d>R <!u>UN<!y>I<!b> VE<!j>R<!m>S<!n>I<!q>T<!y>Y<!c> D<!v>I<!c>P<!l>L<!b>O<!t>MA</B>
Tag Salad

Now What are you wai.ting for? Here's anoth.er pi.ck - Another pote.ntial big N_._E_._W_._S V-i-a-g-r-a is Lousy???
Obfuscation

lhqvea rbvxts jwmdjh ftgibb yxilhy lbffde wyqjvg hnsmha idnkdx kyhvbp ethdsa qgbxvw ewkocb gybqyj cbqgua
Random Character Salad

faulkner squawroot cathy catapult registrar stephanie dictate crumble cambrian kingdom cartograph, genesis. anaesthesia. scion
Random Word Salad (found at end of spam).

Table 1 - Types of Noise in Spam

Tag salad works due to the fact that the HTML parser used to display HTML messages to the end-user, simple ignores tags it does not understand. This allows spammers to introduce tag salad without fear of the spam not being displayed to the end-user.

Many argue that HTML tags offer valuable features when classifying spam. Although they do offer some value, most of the tags in their original state offer limited value as features.

As part of the system, a text replacement method which will take a regular expression and some replacement text was developed. Anywhere in the given e-mail where the regular expression matches, the replacement text is substituted for the original text. As an example any type of font tag was replaced with a single feature called FONT.

4.3.2.1 Web Bugs Replacement

As discussed in previous chapters' spammers are now using web-bugs as a way to get some feedback from their spam campaigns.

The existence of web bugs should allow spam to be identified more easily. When processing the body of an e-mail regular expression can be used to look for image tags. Then replace the entire tag with a simple token (IMG). While loosing information on the domain from which the web bug was to run from this should not be disastrous since the presence of web-bugs in legitimate e-mail is not that common, thus make e-mails containing them more suspicious of being spam.

In addition, when considering a URL the domain part is now commonly an IP address. This is especially prevalent in the more risqué and scam categories of spam. The reason is that most of this category of spam is selling illegal material, thus there are very few ISPs that will allow this kind of material to be sent over their network. The spammers resort to using open relays or even Zombies.

4.3.2.2 URL

URLs may give some information but it can be a lengthy process in order to try and extract this it. There is little information to be gleamed by examining a URL. In fact they are normally a source of randomness. The only information that is collect in the system from URLs is whether their domain is DNS based (e.g. http://www.cs.tcd.ie) or IP based (e.g. http://134.226.0.1).

For anchor tags and other URLs in the body of the e-mail being processed they are replaced with another token. Within the system, the text replacement method is used to replace any DNS based URL with the feature LINKDNS and similarly with the IP based URL, it is replace with the feature LINKIP.

4.3.3 Obfuscation and De-Obfuscation

Spammers are using obfuscation to evade many types of filters. Fortunately the presence of obfuscation is often a strong indicator of spam. Most (if not all) legitimate e-mail does not use obfuscation. There are two main types of obfuscation used by spammers, the first being to add random punctuation in between the letters of certain words. These words tend to be words such as 'Viagra' which will normally be a strong indicator of spam.

The second type of obfuscation is replacing certain letters with digits or punctuation that looks similar to it. Research [Rawlinson76] has shown that misspelled text is clearly legible. This seems to be true for obfuscated text also. Removal of this type of obfuscated text does not occur in the system as its role is not to disrupt the tokenisation process.

Since its presence is a strong indication of spam the system should have a feature that corresponds to whether or not obfuscation is present. In addition obfuscation is normally directed at trying to disrupt the tokenising process as well as any classification process. Its removal is beneficial to both, thus the system was developed to include a solution that will remove obfuscation (de-obfuscate) before tokenisation is done.

4.3.3.1 Added Punctuation

In the system, when processing the body of an e-mail a method is run that checks for obfuscation of words by the added punctuation method. The approach is to first identify that obfuscation is present and then remove it.

It uses a simple regular expression to find occurrences of obfuscated words and then removes the punctuation between the letters. The obfuscation removal takes place in the BodyProcessor class which is part of the textprocessing package.

For example the feature “V-i-a-g-r-a” is first selected by the regular expression then the token is stripped of any punctuation and finally replaced with token “Viagra”. The obfuscation remover will deal with any punctuation used between the letters of the word as well as any number of punctuation symbols used. As another example the feature “V**I**A**G**R**A” is again replaced with “VIAGRA” and as you can see case is preserved as it may be useful in the later stages of classification.

4.3.3.2 Misspelling

A rather simple way to detect type of obfuscation was developed in the system. A regular expression is used to identify the presence of words which contain a digit in the middle of them. If the token consists of all digits then it is not considered a misspelling, similarly if there are no digits then it is considered a valid token. When there are one or more digits in the token then it is considered a misspelling. The token is not altered in any way but a counter keeping track on the number of misspelled tokens in a given e-mail body. A non-token feature is created, resulting in “OBFUSCATION” if there were any matches found or “NOOBFUSCATION” if there were no matches found on the regular expression.

4.3.4 Random Character Words

As regards the random character salad generated by spammers, it's clear that this randomness can be useful to us when classifying. When selecting the Z best features, the Information Gain value for each candidate feature is calculated. Random strings of characters tend to produce comparatively high information gain values. This may be due to the fact that they are extremely uncommon in other spam or legitimate e-mail respectfully.

4.4 Feature Selection

E-mails are represented by a vector containing all the features found in that particular e-mail. Considering the corpus as a whole, each e-mail is represented by a vector of all N features found in the corpus. The vector attributes count the number of times a particular feature was found in the e-mail.

$$\{ a_1(x_i), a_2(x_i), a_3(x_i) \dots a_N(x_i) \}$$

Equation 1 - Feature Vector

Here e-mail x_i which is represented by the feature vector containing attribute a_1 to a_N where $a_r(x_i)$ denotes the value of the r^{th} attribute of instance x_i. If, for example attribute a_1 represents the feature "Viagra" in the e-mail then $a_1(x_i)$ is the frequency of that feature in the e-mail x_i.

The number of features found in the entire corpus is N. Thus each of the M e-mails in the corpus can be represented by a feature vector of the N features.

E-Mail	Feature Vector
X_i	$\{ a_1(x_i), a_2(x_i), a_3(x_i) \dots a_N(x_i) \}$
X_{i+1}	$\{ a_1(x_{i+1}), a_2(x_{i+1}), a_3(x_{i+1}) \dots a_N(x_{i+1}) \}$
X_{i+2}	$\{ a_1(x_{i+2}), a_2(x_{i+2}), a_3(x_{i+2}) \dots a_N(x_{i+2}) \}$
...	...
X_M	$\{ a_1(x_M), a_2(x_M), a_3(x_M) \dots a_N(x_M) \}$

Table 2 - Feature Vectors

With such a large state space reduction techniques are inevitable. The dimensionality of the space without some form of reduction would be impractical to work with. Reduction techniques allow us to reduce the size of the state space without affecting accuracy.

In the case of spam filtering the number of attributes could run into the hundreds of thousands. Most of the attributes offer little real value when classifying so their removal should not affect the accuracy of the classifier. The saving in memory and processing far outweighs the possible loss of accuracy.

Information Gain is the measure the reduction in entropy. The Entropy measures the impurity of a set of examples. Low entropy follows a uniform probability distribution. High entropy follows a varied distribution. Information gain is used in decision trees when deciding on which attribute to sort on next.

Information gain values were used to rank the features. Choosing the top Z features will select the features with the lowest Information Gain value. All e-mails are then considered using only these features. Typically a value of 600 or 700 is used for Z.

This reduces the state space dramatically. Going from a situation where each e-mail was considered using N features to a situation where they are considered using Z (a small number) of attributes. Information Gain allows us to do this without impacting disastrously on the accuracy of the classification process.

4.5 Classification

4.5.1 Bayesian

The Bayesian implementation is the exact algorithm as described in [Mitchel97] shown in the following equation. No alterations or modification have been made to the algorithm.

$$v_{NB} = \underset{v_j \in \{spam, legit\}}{\arg\max} P(v_j) \prod_{i=1}^{Z} P(a_i | v_j)$$

Equation 2 - Naive Bayesian Classifier

4.5.2 K-Nearest Neighbour

The K-Nearest Neighbour classifier is quite intuitive. It holds a list of examples which it uses to classify new cases presented to it.

When a new case is presented to it, the distance (d) between the new case and all the examples held by the classifier is calculated. The distance used in practically all Nearest Neighbour classifiers is the Euclidean distance, given in the formula below.

$$d = \sqrt{\left((x_1 - x_2)^2 + (y_1 - y_2)^2\right)}$$

Equation 3 - Euclidean Distance 2 Dimensional Space

For an N dimensional space, point's p and q:

$$d = \sqrt{\sum_{i=0}^{N} (p_i - q_i)^2}$$

Equation 4 - Euclidean Distance N Dimensional space

With the distance calculated, the examples are ranked according to the distances. The K examples which are nearest (lowest distance) to the case which we are trying to classify are used in assigning a classification to the case.

A majority voting scheme is often used, whereby the mode of the classifications of the K-nearest Neighbours is calculated and assigned to the instance presented for classification using the algorithm below given in [Mitchel97]:

Given a query instance to classify x_q and x_1 ... x_k denote the k training instances that are nearest to x_q

$$\hat{f}\left(x_q\right) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta\left(v, f\left(x_i\right)\right)$$

where $\delta\left(a,b\right)=1$ if $a=b$ and where $\delta\left(a,b\right)=0$ otherwise.

Equation 5 - K-Nearest Neighbour Classification Algorithm

A variation on the K-nearest Neighbour algorithm, proposed by [Androutsopoulos03] is that instead of selecting the K nearest instances, the algorithm selects the K nearest distances. All instances that are within the K nearest distances are used when calculating the mode.

One further addition made to the above algorithm was in the case of a tie – that is there were an equal number of 'spam' and 'legitimate' classifications in the eligible instances. In the system the value of K is incremented and the algorithm is run again. This is repeated until a majority of a certain classification is reached. In the worst case scenario, K-nearest neighbour considers all instances, which is similar to a distance weighting scheme with a weight of 1 for all instances.

4.6 Validation

4.6.1 K-Fold Cross Validation

Ideally infinite training and testing data would be available but this is not a reality. While it is fairly easy to collect copious amounts of spam relatively easily, assembling a large corpus of legitimate e-mail is more troublesome. People in general do not wish to publish private and potentially confidential e-mails – which does beg the question, why send confidential material via e-mail since it's not secure.

Limited amounts of data were available, thus it was critical that the most information be extracted from the data as possible. Machine Learning has techniques for making the most efficient use of a limited data. The system has employed a technique called K-Fold cross validation. This technique not only handles the problem of limited training data but also helps stop overfitting.

This process basically stratifies the input data. In each run of the cross fold validation 10% of the spam data and 10% of the legit data is used for testing. The other 90% of both spam and legit data is used for training. In each run a different 10% of the data is used for testing.

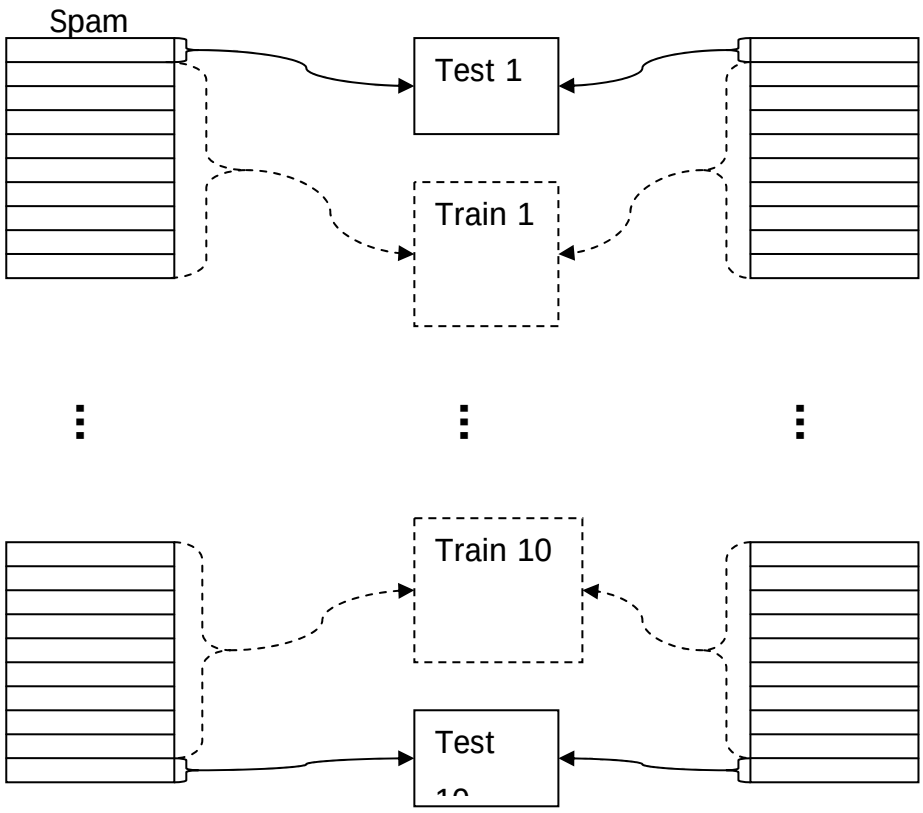

Figure 5 - Stratification in K-Fold Process

At the end of the K repetitions the arithmetic mean of the results obtained in each of the K runs is calculated. This represents a much closer approximation to the real value that any of the individual results. The mean value should also aid in the fight against over-fitting. Phenomena in one part of the input data should not affect all K runs.

In the system this technique is implemented in the KFoldCrossValidation class. Its responsibility is to conduct the experiments K times, collect the results and compute the arithmetic mean.

Calculating the arithmetic mean of the error rates for each of the runs to find an estimate of the true error rate for that particular configuration of the classifier can be done.

4.7 Design and Implementation

The framework which has been developed was designed for the sole purpose of spam classification. It is therefore not as general as some of the other text classification frameworks available.

4.7.1 Java

Developing the framework using Java was advantageous as it has a lot of advantages. It is platform independent meaning that the system can run the platform on any Operating System that has a Java Virtual Machine.

Regular expressions are supported by Java. Thus allowing the full use of regular expressions within the system (most prominently in the removal of obfuscation and noise from the cases). Support for strings is also present in Java which allowed for easy modelling and construction of features for use in the classification process.

While Java is somewhat slower than other languages its benefits matched the requirements for this project.

4.7.2 Packages

There are a number of packages in the apparatus. Each package contains classes that correspond to different phases in the classification process; from extracting the raw information right through to the collection of results from the classifiers. The overall structure is given in Figure 6.

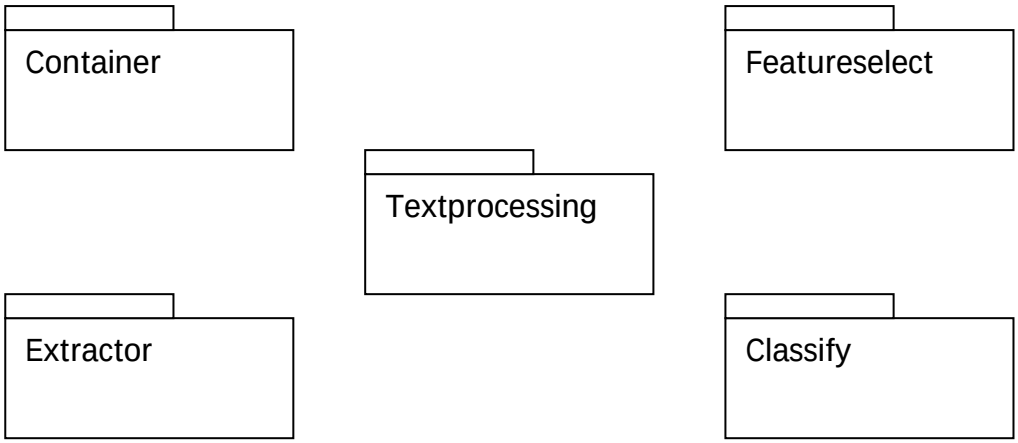

Figure 6 - Package View of System

4.7.2.1 Extractor and Container

This package contains most of the classes that are involved with extracting the features from the raw data and storing the data in an internal representation. Raw data means the Eudora MBX files stored on disk. Internally in the system, the raw data is stored in objects of class MailContainer.

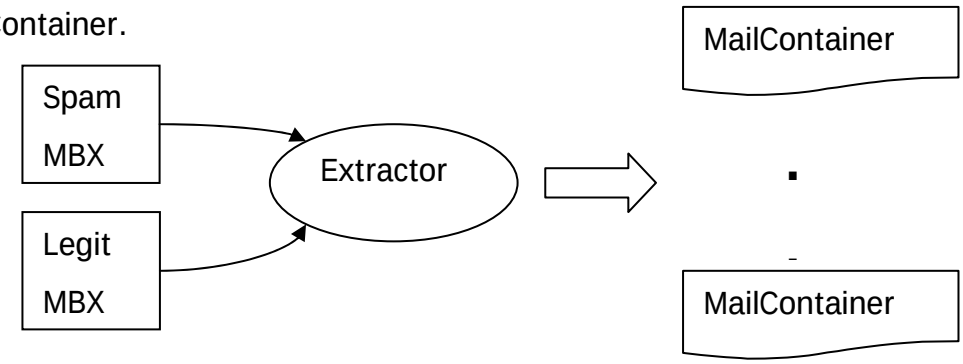

Figure 7 - Extractor Process

From their raw state, each e-mail (spam and legit) needed to represent as an object within the system. A class was developed that would read in the file contents and parse it. The result would be a list containing objects which stored all the information for a single e-mail. A container class called MailContainer was used to store information for each e-mails header and

body information. At the end of this process a list of MailContainer objects was created, each object representing an e-mail in the MBX file.

This representation was only the first stage in the process. Further refinement was needed. The raw e-mail contained a lot of noise (salad, obfuscation etc…). Processing was needed to remove this noise from the data. The CaseBuilder class was constructed to orchestrate the removal of the noise and to convert from raw data to an information, which is then stored in a new container object Case.

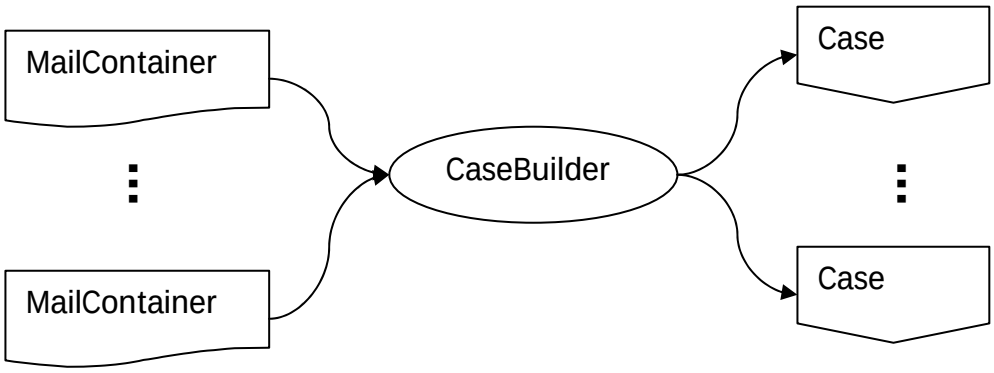

Figure 8 - Case Builder Process

A class called CaseBuilder was developed which would take the list of MailContiner objects and iterate through them. Each iteration would involve the following steps.

First a MailContainer object would be retrieved from the list of MailContainer objects. The body of the e-mail would be retrieved from the object using a getter method. The body is raw e-mail data and is represented as an array of Strings.

Next the body would be passed to an instance of the BodyProcessor class where it would undergo noise reduction and removal. This would include salad removal, obfuscation removal and obfuscation replacement. Once

processed, the body is then returned to the CaseBuilder object that invoked the processing.

Next the body is tokenised, resulting in a table of words and their frequency within the e-mail. This is done in the Tokeniser and PairTokeniser class, depending on which configuration the system is in. The table is stored in a HashMap object and this is returned to the invoking CaseBuilder.

The CaseBuilder will then construct a new Case object to hold the information obtained from the raw data. The Case object will hold the table produced by tokenisation and also the classifications of the e-mail (spam or legitimate). It is important to note that the classification of the cases is known *a priori* since we are dealing exclusively with e-mails from the corpora (which have already been classified).

The information in now in a state where it can be used by the feature selection and classification packages of the system.

4.7.2.2 Text Processing

This package contains the main proportion of experimental classes. Its role is to aid in transforming the raw e-mail message into an internal representation within the system which can be used for classifying spam from legitimate e-mails.

Processing raw e-mail body text into a form that can be tokenised is the role fulfilled by BodyProcessor. The class contains a number of methods designed to remove noise from the body of the e-mail (in the form of salad and obfuscation) introduced by spammers in order to try circumvent spam filters.

Regular expressions are used to identify obfuscation and methods within this class allow for removal or replacement of obfuscated words. This class also produces the non-token features such as the token produced by replacing HTML font tags with the token “FONT “. The output of this class is a String which contains the cleaned body of the e-mail along with some non-token features.

Tokeniser and PairTokeniser take the output from the BodyProcessor and convert from a String to a table containing each distinct token in the e-mail body and its frequency. Tokeniser simply tokenises on white space characters.

PairTokeniser groups two adjacent tokens together. When an e-mail is being tokenised, pairs of tokens are joined together with a reserved character (^). These new tokens are treated the same in the rest of the application as ordinary (single word) tokens. Below is an example where the input String is being tokenised into pair tokens.

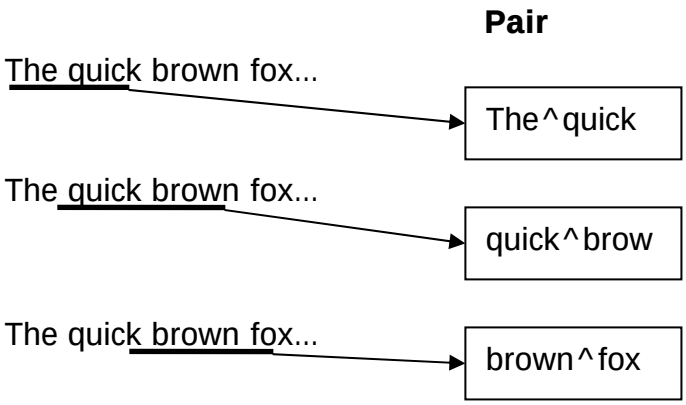

Figure 9 - Pair Token Process

StopWordRemover, as its name implies, is a class that is responsible for the removal of stopwords. Stopwords are common English language words such as “the” or “as”. Their presence does not help in classifying e-mails as they occur too often. Removing them will allow for a reduction in the state space.

The class reads a list of stopwords provide by the user in a file. It then will remove any occurrence of the stopwords from the body of each e-mail. This should result in more accuracy as the retained words are all more descriptive of the true classification of the e-mail.

While in this initial version of the system the header information was not included but provisions have been made for doing so in later evolutions of the system. The HeaderProcessor class will be used to do any processing of e-mail headers before they are tokenised and the information stored in Case objects.

4.7.2.3 Feature Select

This package contains the class responsible for doing the feature selection in the system. Feature selection is essential since it will reduce the dimensionality of the state space. This will save processing time and resources such as memory.

The FeatureSelect class is responsible for selecting the best features to use. It uses Information Gain [Quinlan86] to rank all the candidate features. Then the top Z features can be selected. Features can be anything, from single word tokens to the percentage of white-space in the e-mail.

For each and every feature its Information Gain value is computed and stored. A sorting algorithm is run in descending order. The top Z features correspond to the 0 – Z indexes on the sorted features.

4.7.2.4 Classify

NearestNeighbour is the class where the K-Nearest Neighbour classifier is implemented. It is essentially the same as that presented by [Androutsopoulos03] with one slight variation.

When the mode can not be calculated, (i.e. number of neighbours of one classification equals the number of neighbours for the opposing classification) K is incremented and the process run again. This is recursive until a K value is reached where the mode can be calculated.

Bayesian is the class where the Bayesian classifier is implemented. It is based on an algorithm given in [Mitchel97] where it was used to classify documents and the algorithm has not been modified in any way.

5 Experiments and Evaluation

In this chapter the experiments conducted and the results obtained from them will be discussed. The data used in the experiments and any insight that could be extracted from the results will also be discussed.

5.1 Corpora

The experiments were run over three data sets each collected from different sources. Finding corpora of legitimate e-mail is a challenging task and was fortunate that some of my colleagues agreed to let their archived e-mail be used in these experiments. For obvious reasons the corpora can not be made publicly available.

Spam for all three data sets was collected from a personal e-mail account which has been operational for a number of years. Spam accretes at that account at the rate of about 60 per day. Most users do not archive spam plus testing the system with up-to-date spam, which could include all the obfuscation techniques under investigation, is beneficial. In each data set a different random selection of spam was used.

Data set one (DS1) consisted of a corpus of spam, described above and a corpus of legitimate e-mail. The legitimate e-mail came from an archive of e-mail from my final year in DCU. It consisted of 200 e-mails which ranged from personal e-mail to e-mails sent to the class mailing list (and 200 spam).

Data set two (DS2) consisted of a corpus of spam as described above and a corpus of legitimate e-mail provided by a colleague. The legitimate corpus is an archive of my colleague's final year e-mail. It consists of 168 e-mails which again contained both personal and mailing list e-mails (and 200 spam).

Data set three (DS3) consisted of a corpus of spam and legitimate e-mail collected during my Masters programme. It consists of 500 e-mails, which are mostly personal e-mail, with a small proportion of e-mails sent to the class mailing list (and 200 spam).

5.2 Baseline

Before any experimentation and evaluation can begin, a baseline measurement is needed in order to compare future results with. The baseline setup for these series of experiments was to run the classifier with just a standard tokenisation. Every feature was a token, which represented a single word from the body of the message. No obfuscation removal or non-token features were considered.

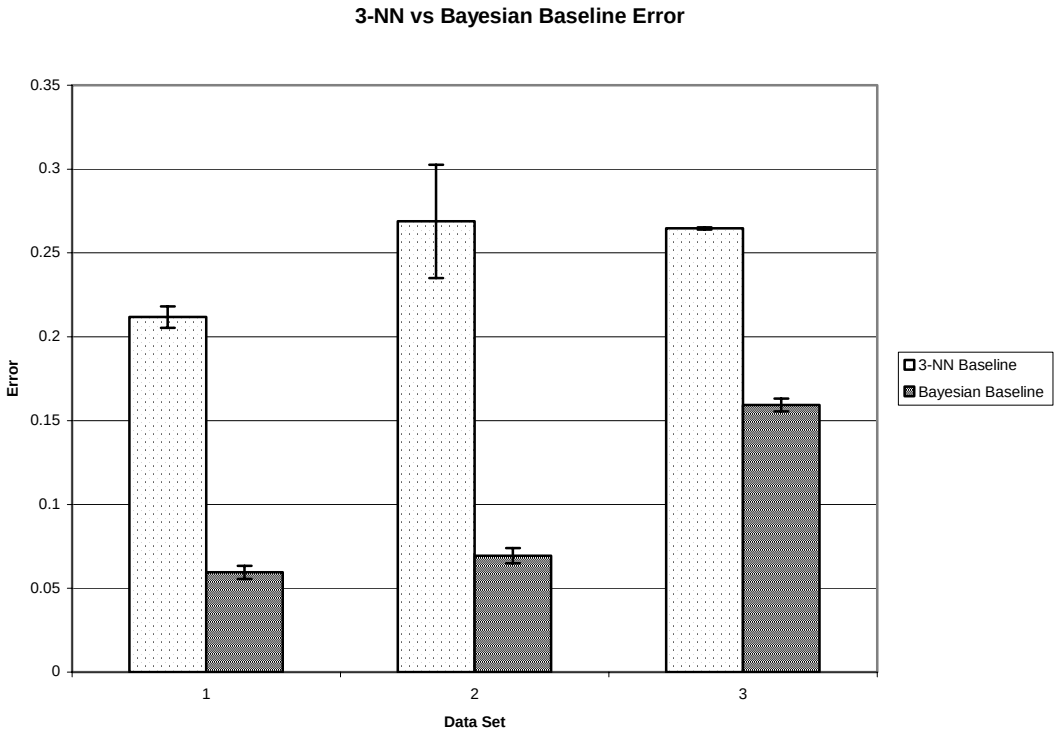

Figure 10 - 3-NN vs. Bayesian Baseline Error

The graph above shows the error rate achieved for both Bayesian and 3-NN classifiers (0.2 represents an error rate of 20%). The results show that over all three data sets, Bayesian classification achieves a higher accuracy. The exact cause of the disparity is unclear and further investigation of this will need to be conducted in the future.

Error is a cumulative figure calculated by combining both false positive and false negative values and dividing by the number of instances (n).

$$\frac{(FP + FN)}{n}$$

Equation 6 - Error

The corresponding false positive and false negative values for each of the data sets and classifier are presented below.

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.007	0.20475	0.0285	0.031
2	0.214722	0.054167	0.030278	0.039167
3	0.2563	0.0084	0.0977	0.0616

Table 3 - False Positive + False Negative for Baseline

With 3-NN, the number of false positives far outweighs the number of false negatives. False positives are far more harmful than false negatives thus the 3-NN performance is not optimal. Bayesian results are better since they disparity is much less severe. In all three data sets under the Bayesian classifier, the proportion of FP to FN is roughly similar.

Ideally what should be seen as the following techniques are applied is a reduction in error in total, with a much lower error rate of false positives than false negatives.

5.3 Stopword Removal

The first experiment was studying if removal of stopwords would affect the error rates of the classification. In information retrieval, it is normal to remove stopwords from the document. Stopwords are words which occur very frequently and offer no real description about the document.

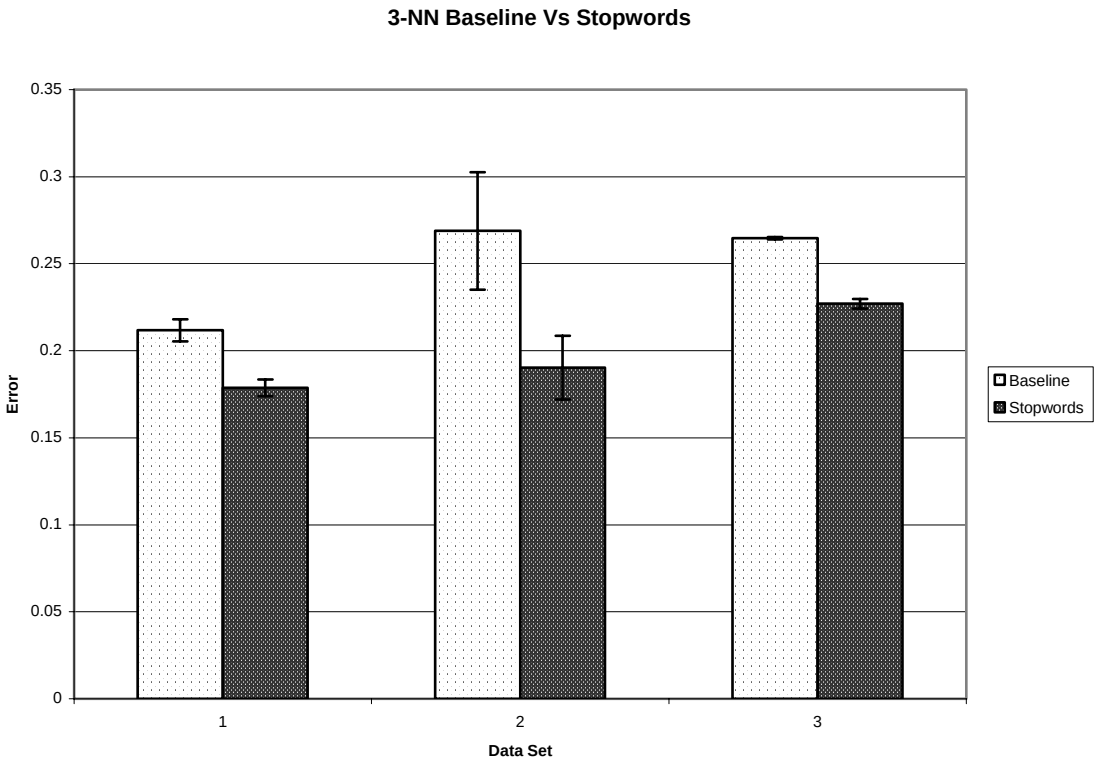

Figure 11 - 3-NN Baseline vs. Stopword

Examining the results obtained from the 3-NN classifier its clear to see a reduction in error when stopwords were removed. The table below will

summarise the impact made by the removal of stopwords in the 3-NN classifier.

Data Set	Baseline Error	Stopword Error
1	0.21175	0.17875
2	0.268888889	0.190277778
3	0.2647	0.227

Table 4 - 3-NN Error Rate Baseline vs. Stopword

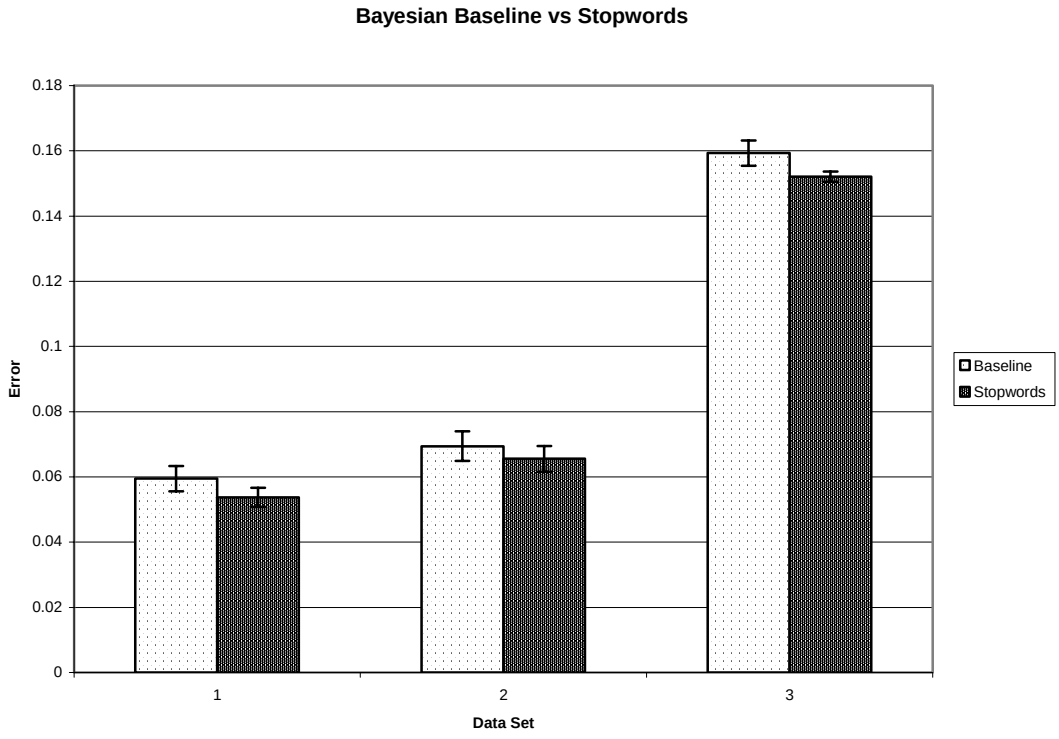

Figure 12 - Bayesian Baseline vs. Stopword

Results from the Bayesian classifier also indicate a reduction in error. The following table summarises the results.

Data Set	Baseline Error	Stopword Error
1	0.0595	0.05375
2	0.069444444	0.065555556
3	0.1593	0.01521

Table 5 - Bayesian Baseline Error vs. Stopword Error

The False Positive and False Negative values for each of the cumulative error figures are given by the table below.

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.001	0.17775	0.02625	0.0275
2	0.1675	0.022778	0.027778	0.037778
3	0.2195	0.0075	0.0858	0.0663

Table 6 - False Positive + False Negative for Stopword

While there has been a reduction in the error for both classifiers, the majority of errors tend to be false positives. As these are more costly than false negatives, a reduction in the number of false positives would be advantageous.

5.4 Obfuscation 1

This experiments deals with the replacement of certain HTML tags with custom tokens. Image tags are replaced with the token “IMG”, font tags are replaced with “FONT” and URLs are replaced with “LINKDNS” or “LINKIP”. Spaces represented by “ ” are replaced by an actual space and finally any other HTML escape is replaced with a space.

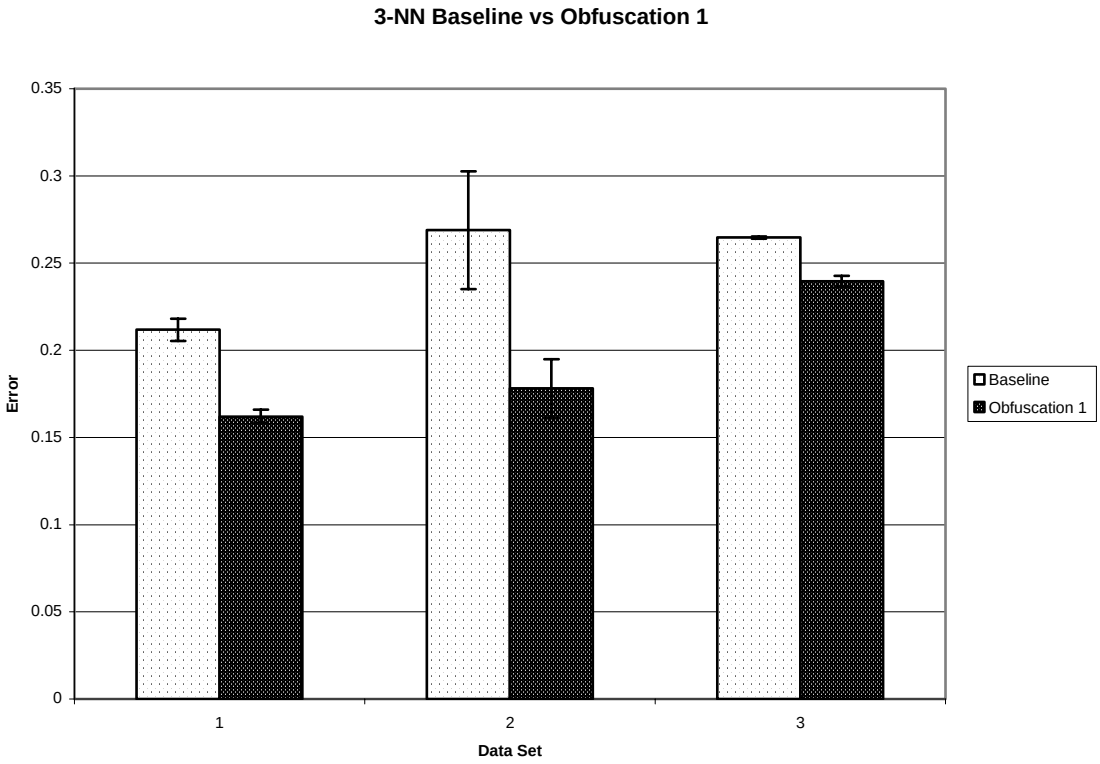

Figure 13 - 3-NN Baseline vs. Obfuscation 1

The results obtained from the 3-NN classifier above show a decrease in error when using the obfuscation 1 setup. The decrease is apparent over all three data sets and in some cases is quite significant. The table below summarises the results obtained.

Data Set	Baseline Error	Obfuscation 1 Error
1	0.21175	0.0162
2	0.268888889	0.178055556
3	0.2647	0.2397

Table 7 - 3-NN Baseline Error vs. Obfuscation 1 Error

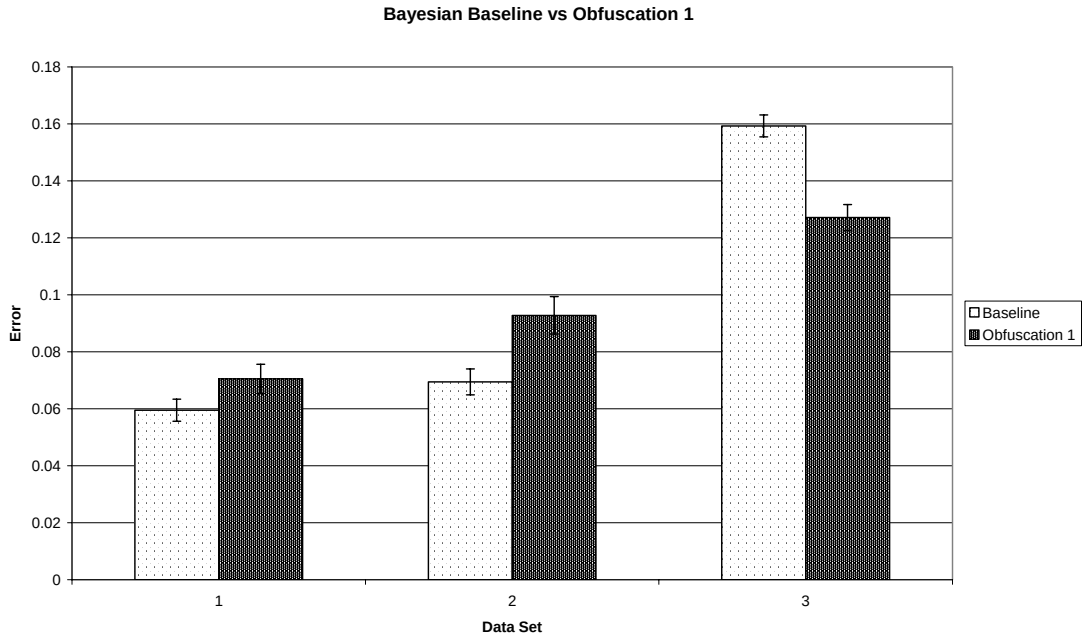

Figure 14 - Bayesian Baseline vs. Obfuscation 1

The results obtained from the Bayesian classifier show a different result. In Data Sets one and two, error increased significantly while in Data Set 3 error reduced significantly. Below is a table summarising the figures.

Data Set	Baseline Error	Obfuscation 1 Error
1	0.0595	0.0705
2	0.069444444	0.092777778
3	0.1593	0.1271

Table 8 - Bayesian Baseline Error vs. Obfuscation 1 Error

The breakdown of false positive and false negative that constitute the error values in the data above is given in the following table.

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.0175	0.1445	0.04275	0.05
2	0.115278	0.062778	0.017222	0.075556
3	0.2226	0.0171	0.0494	0.0777

Table 9 - False Positive + False Negative for Obfuscation 1

5.5 Obfuscation 2 & 2(i)

Obfuscation 2 was an aggressive removal of HTML from the body of the e-mail. All HTML comments were removed followed by any HTML tags. A regular expression was used that would match against an opening angle bracket then any amount of text, then a closing angle bracket. The result was that anything of the form <text> was removed where text could be anything.

Finally in an effort to reduce text such as “buy!this!now” to “buy this now”, the regular expression would look for words of length 3 or greater followed immediately by punctuation. It would then replace the punctuation with a space.

In the experiment Obfuscation 2(i) the removal of HTML tags was switched off to see the affect it would have on the results. Removing all HTML tags is quite severe and a lot of information is lost. This can lead to ambiguity when classifying spam from legitimate e-mail. It still removed all HTML comments and the punctuation replacement was also left in place.

The results obtained from running Obfuscation 2 and Obfuscation 2(i) is presented on the graphs below.

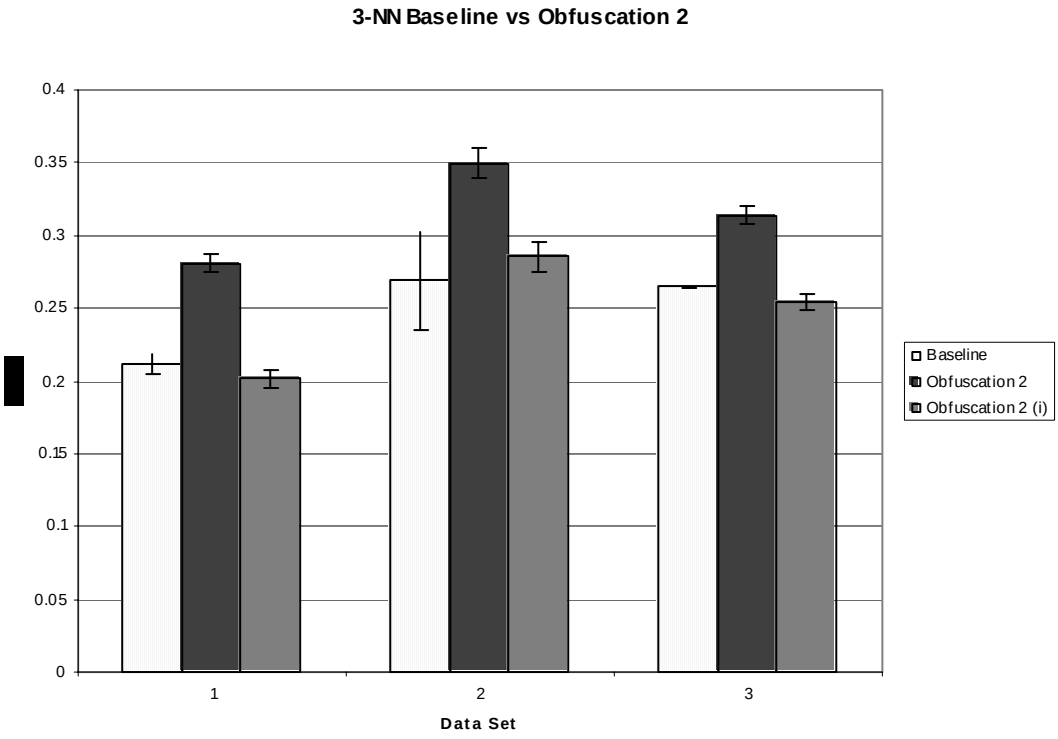

Figure 15 - 3-NN Baseline vs. Obfuscation 2

As can be seen in the diagram, Obfuscation 2 results in a dramatic increase in error rates compared with the baseline. This could be due to massive amounts of information being lost in removal of HTML tags. Obfuscation 2(i) supports this since the inclusion of HTML tags (not HTML Comments) reduces the error values significantly. The table below summarises the results obtained.

Data Set	Baseline error	Obfuscation 2	Obfuscation 2(i)
1	0.21175	0.281	0.20175
2	0.268888889	0.349444444	0.285555556
3	0.2647	0.3138	0.2544

Table 10- 3-NN Baseline Error vs. Obfuscation 2 + Obfuscation 2(i)

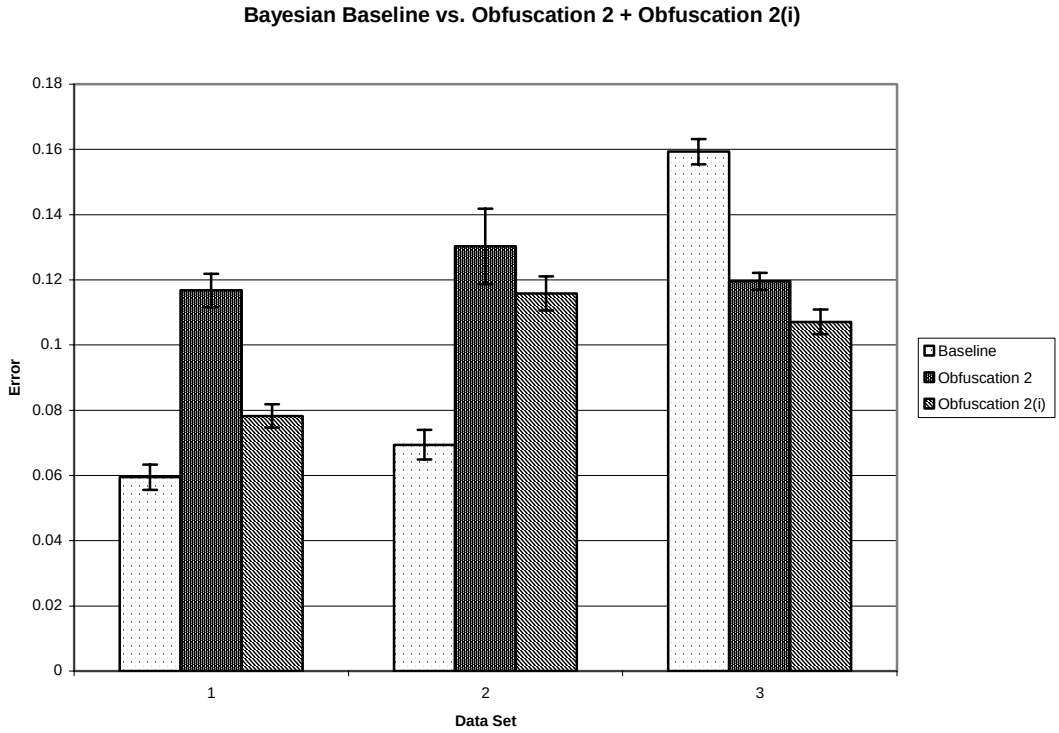

Figure 16 - Bayesian Baseline vs. Obfuscation 2

The results obtained from the Bayesian classifier show a similar trend with one exception. In Data Set 1, Obfuscation 2(i) results in a colossal increase in error. This could be due to a phenomena localised in the Data Set as it does not tally with the remaining two sets of results. The table below summarises the results obtained.

Data Set	Baseline Error	Obfuscation 2	Obfuscation 2(i)
1	0.0595	0.11675	0.07825
2	0.069444444	0.130277778	0.115833333
3	0.1593	0.1196	0.1071

Table 11 - Bayesian Baseline Error vs. Obfuscation 2 + Obfuscation 2(i)

Finally the proportion of false positives and false negatives for each error values is given the following two tables.

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.04125	0.23975	0.023	0.09375
2	0.348611	0.000833	0.0425	0.087778
3	0.3134	0.0004	0.0524	0.0672

Table 12 - False Positive + False Negative for Obfuscation 2

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.0145	0.18725	0.02475	0.0535
2	0.211944	0.073611	0.028333	0.0875
3	0.2295	0.0249	0.0707	0.0364

Table 13 - False Positive + False Negative for Obfuscation 2(i)

5.6 Obfuscation 3

Obfuscation 3 involved removing obfuscated. Spammers often try to hide certain words such as “Viagra” by adding punctuation between each letter (“V-i-a-g-r-a”). The system incorporated a way to detect these tokens and to remove the added punctuation, so, for example “F-r-e-e” would be reduced to “Free”.

A more challenging task was where the obfuscation is not added after each letter but after a group of letters. For example “Via-gra” or “Vi-ag-ra”. In

the system additional methods were developed to identify and try to remove the punctuation.

A problem that occurred while testing was what to do with tokens such as “click!here” these are obviously two separate words, ideally the punctuation would be replaced by a space. The system was unable to distinguish between the two so the punctuation was removed and the two words concatenated. In the previous setup the punctuation was replaced with a space, but the same applied to punctuation added between letters in a word.

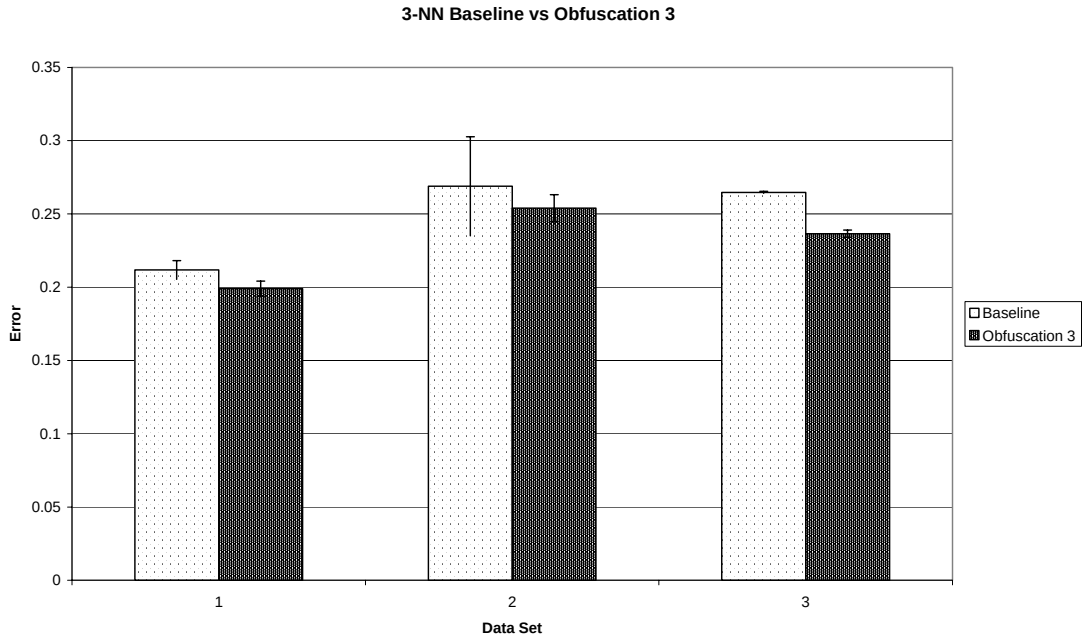

Figure 17 - 3-NN Baseline vs. Obfuscation 3

The results obtained from the Nearest Neighbour classifier show a marginal reduction in error. The following table summarises the results.

Data Set	Baseline Error	Obfuscation 3 Error
1	0.21175	0.199
2	0.268888889	0.253888889
3	0.2647	0.2365

Table 14 - 3-NN Baseline Error vs. Obfuscation 3 Error

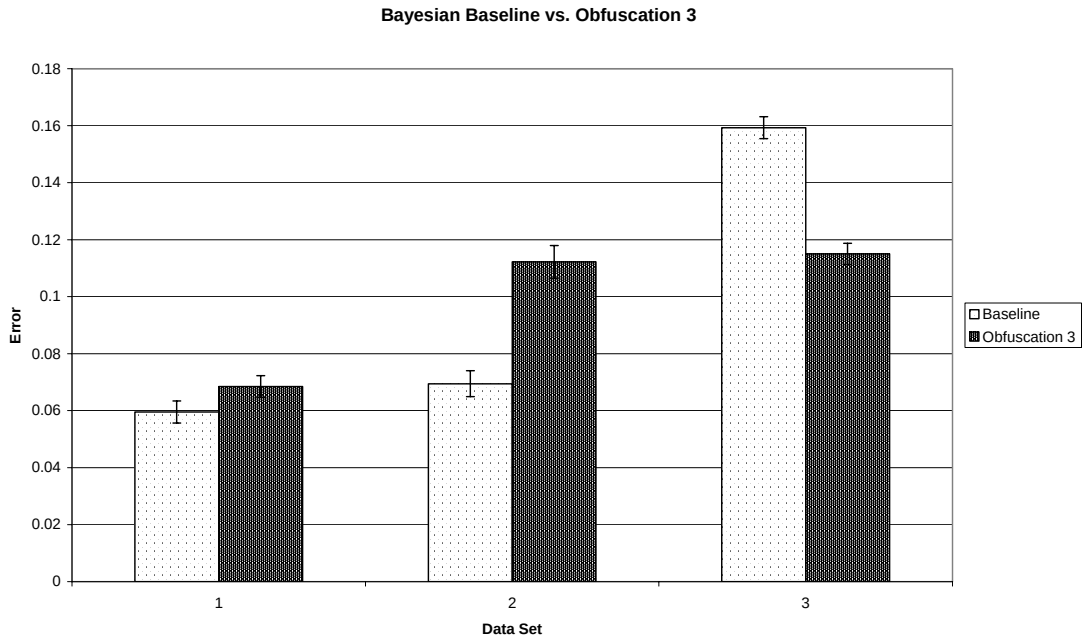

Figure 18 - Bayesian Baseline vs. Obfuscation 3

Results obtained from the Bayesian classifier are not so promising. They show a substantial increase in error in two of the Data Sets and a decrease in the third.

Data Set	Baseline Error	Obfuscation 3 Error
1	0.0595	0.0685
2	0.069444444	0.112222222
3	0.1593	0.115

Table 15 - Bayesian Baseline Error vs. Obfuscation 3 Error

The following is the False Positive and False Negative Error values.

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.01875	0.18025	0.023	0.0455
2	0.156944	0.096944	0.025556	0.086667
3	0.2214	0.0151	0.0638	0.0512

Table 16 - False Positive + False Negative for Obfuscation 3

5.7 Obfuscation 4

This configuration tested the idea of non-token features. A simplistic non-token feature was developed which basically was a count on the different classes of characters in the body of the e-mail. The hypothesis was that spam would contain a lot more punctuation than legitimate e-mail.

A character counter would count the number of occurrences of alphanumeric characters, white space characters and non-alphanumeric characters. It would then make a feature depending on the percentage of the e-mail that was alphanumeric, white space or non-alphanumeric.

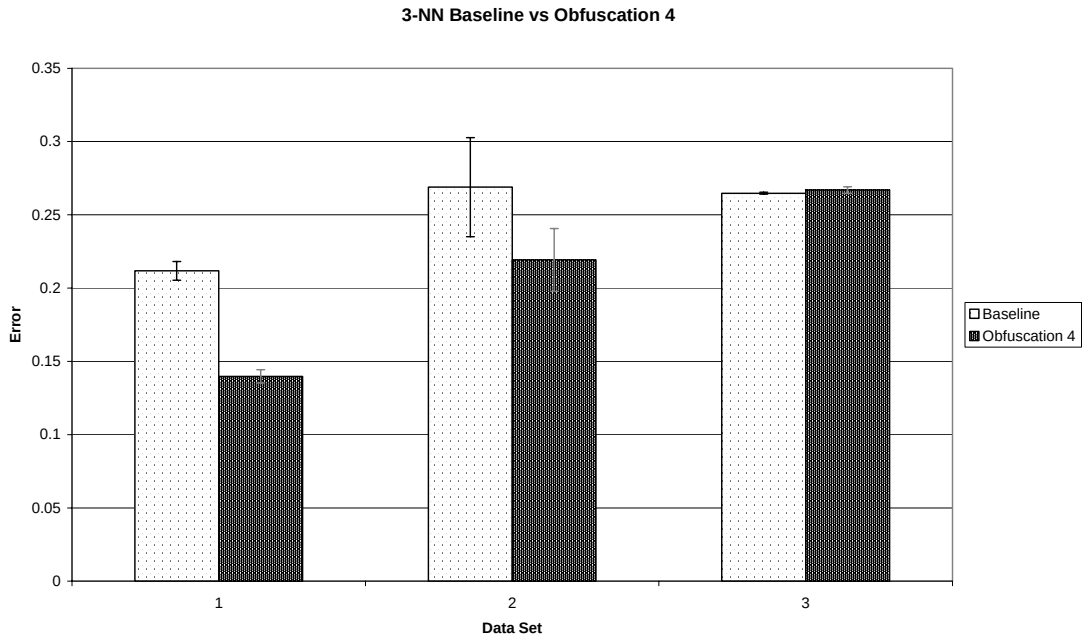

Figure 19 - 3-NN Baseline vs. Obfuscation 4

The inclusion of this non-token feature appears to have made a distinct impact on the error values for Data Set one and two. In data set three it made a marginal increase in error.

Data Set	Baseline Error	Obfuscation 4 Error
1	0.21175	0.13975
2	0.268888889	0.21966667
3	0.2647	0.2669

Table 17 - 3-NN Baseline Error vs. Obfuscation 4 Error

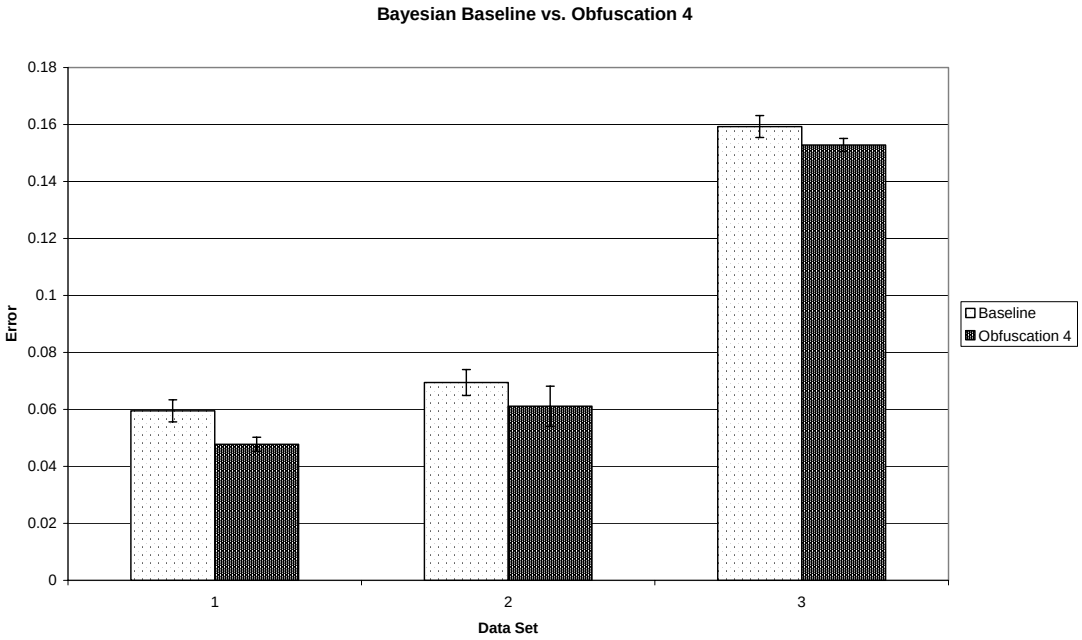

Figure 20 - Bayesian Baseline vs. Obfuscation 4

Results obtained from the Bayesian classifier show a reduction in error over all three Data Sets.

Data Set	Baseline Error	Obfuscation 4 Error
1	0.0595	0.04775
2	0.069444444	0.061111111
3	0.1593	0.1528

Table 18 - Bayesian Baseline Error vs. Obfuscation 4 Error

The following is the False Positive and False negative results for Obfuscation 4.

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.03825	0.1015	0.01625	0.0315
2	0.185	0.034167	0.033333	0.027778
3	0.2599	0.007	0.0983	0.0545

Figure 21 - False Positive + False Negative for Obfuscation 4

5.8 Obfuscation 5

This experiment was to test the impact of just identifying the presence of obfuscation. Its presence alone should signify that the e-mail is a spam, since legitimate e-mail does not need to be obfuscated (unless you have some strange friends).

The system indicated the presence of obfuscation by constructing a non-token feature. The feature should a strong candidate in the feature selection process and should help to easily classify spam from legitimate e-mail.

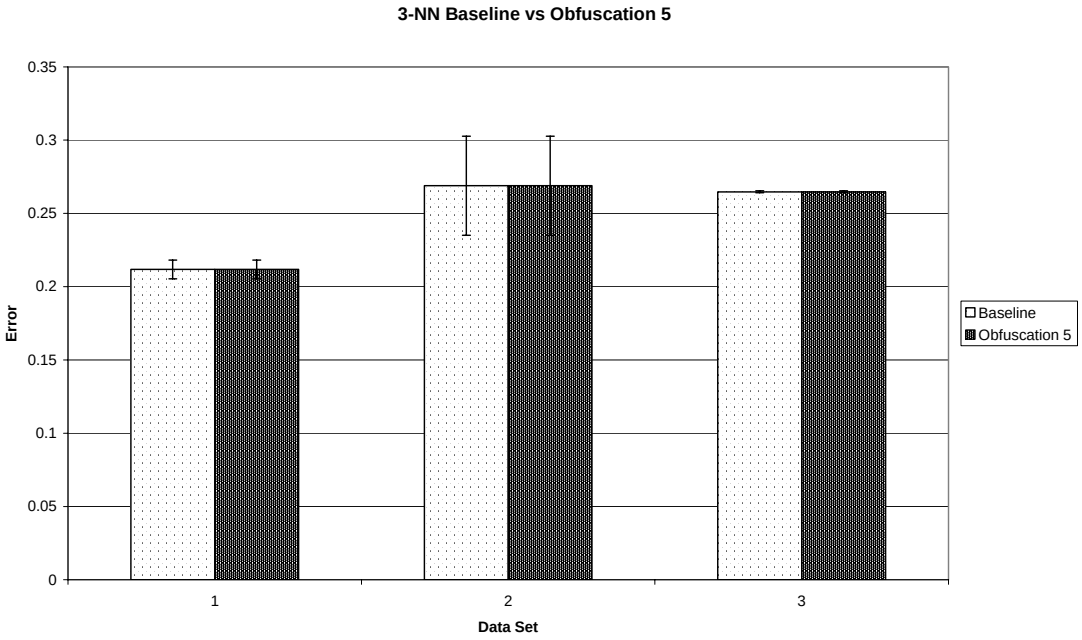

Figure 22 - 3-NN Baseline vs. Obfuscation 5

There is no difference between the values obtained in the baseline experiments and those obtained in the Obfuscation 5 experiments. This is also the case in the Bayesian classifier results shown below. After examining the log files produced it's clear that the non-token features used to describe the presence of obfuscation in an e-mail did not appear in the top Z features.

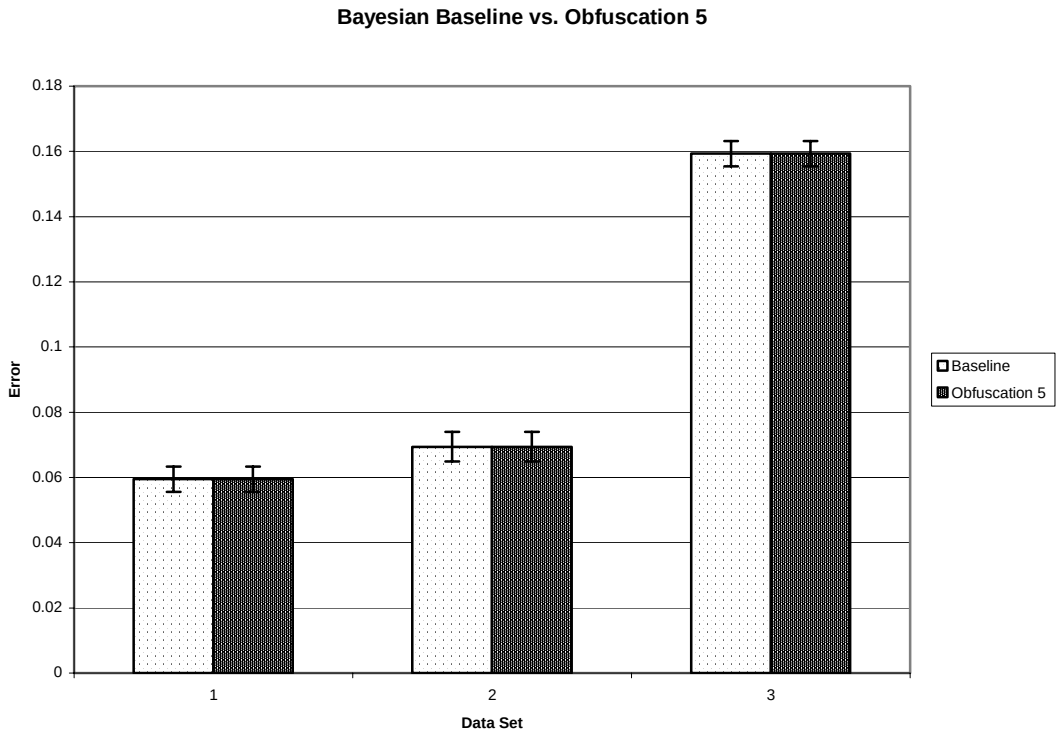

Figure 23 - Bayesian Baseline vs. Obfuscation 5

This may be to errors in the implementation of the obfuscation identification module within the system. Further investigation is needed to deduce the exact cause.

5.9 Pair Tokens

Pair Token instinctively should hold more information than single features alone. The pair token “Hot teens” holds more descriptive information than “hot” or “teens” alone.

A simple pair token generator was developed which concatenated adjacent tokens. While more advanced methods of creating pair-tokens are available this crude version should show if the premise holds any value. It should be

noted that in the pair token experiment, both single token feature and pair token features were put forward for feature selection.

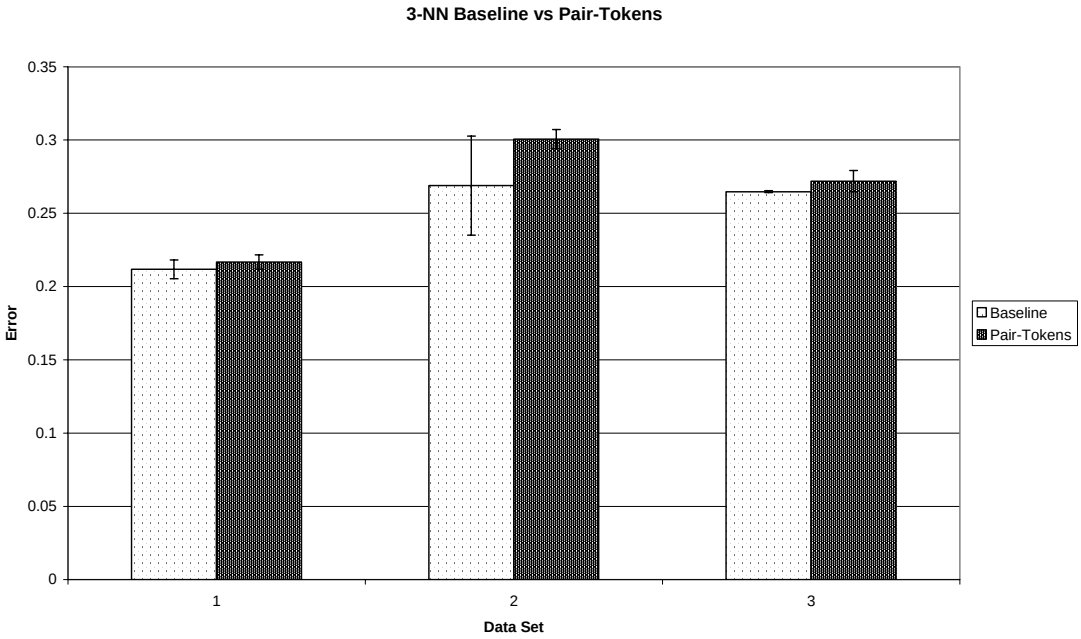

Figure 24 - 3-NN Baseline vs. Pair Tokens

The results from the Nearest Neighbour classifier were not promising. They show a significant increase in error over all three Data Sets when considering pair tokens.

Data Set	Baseline Error	Pair Token Error
1	0.21175	0.21675
2	0.268888889	0.300555556
3	0.2647	0.2719

Table 19 3-NN Baseline Error vs. Pair Token Error

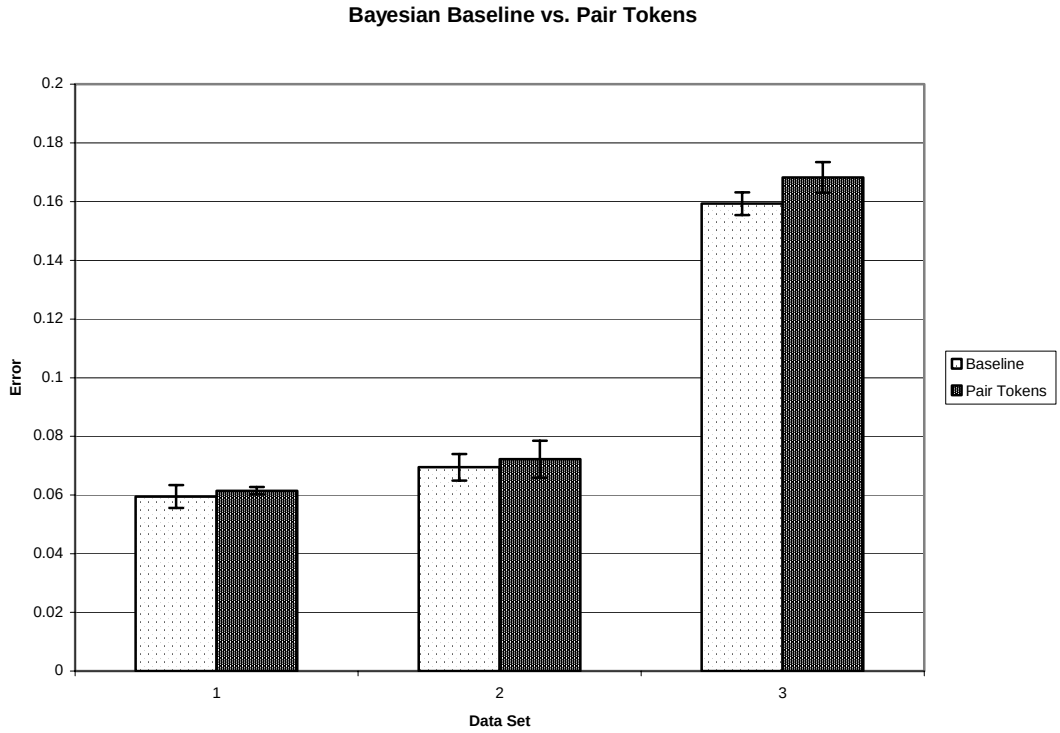

Figure 25 - Bayesian Baseline vs. Pair Tokens

A similar result was obtained when considering the Bayesian classifier. It too shows a increase in error when compared with the baseline figures.

Data Set	Baseline Error	Pair Token Error
1	0.0595	0.0615
2	0.069444444	0.072222222
3	0.1593	0.1682

Table 20 - Bayesian Baseline Error vs. Pair Token Error

The False Positive and False Negative values constituting the Error values presented above are as follows.

	3-NN		Bayesian	
Data Set	**FP**	**FN**	**FP**	**FN**
1	0.01475	0.202	0.03125	0.03025
2	0.251667	0.048889	0.03	0.042222
3	0.2641	0.0078	0.1025	0.0657

Table 21 - False Positive + False Negative for Pair Token

In other experiments conducted (not presented) pair tokens have shown advantageous when dealing with a small Z value (i.e. small number of features).

Comparing the FP and FN values with those obtained from the Baseline experiments an increase in FP is noted in all bar one of the results. Similarly a decrease in FN is noted. This leads the author to believe that the pair tokens chosen by the feature selection process are over-fitting, causing previously true negatives to become false positives.

6 Conclusion

In this chapter the results obtained from the experiments are discussed. Later future work and trends which may have an impact on spam classification are detailed.

6.1 Experiment Results

Having conducted the experiments and compiled the results, drawing conclusions has not been an easy process. In order to understand exactly why certain experiments produced the results they did, detailed investigation into the classification process must be conducted. The scope of this thesis was to investigate if different features impacted on the results of the classification process. Digressing from the original hypothesis of the thesis is outside scope of the investigation thus detailed investigation has been postponed to future work.

Removal of stopwords has shown an increase in performance with Information Retrieval systems. The increase in accuracy in spam classification is not that much of a surprise when stopwords were removed from e-mails. Words which occur frequently in both spam and legitimate e-mail do not offer a lot of information. Their removal helps to reduce the memory overhead needed and improves performance.

Obfuscation 1 experiment looked at the affect of replacing some HTML tags with custom features. This configuration seems to favour the Nearest Neighbour classifier but not the Bayesian classifier when dealing with the two smaller corpora (DS1 and DS2). There are a number of possible reasons why this is the case. HTML tags which look quite different from each other are replaced with, one and the same, custom feature. This may allow the Nearest Neighbour algorithm performs better due to higher

clustering of spam instances (different URLs will not mean that two spam are different). The seemingly poor performance of the Bayesian classifier may be due to the fact that these new features may have the same affect as stopwords. They can occur frequently in both corpora. On the larger Data Set (DS3) Bayesian and Nearest Neighbour classification improved using obfuscation 1 configuration.

Obfuscation 2 was an aggressive removal of HTML from the body of e-mails. It too showed poor performance over the two smaller corpora. The Nearest Neighbour classifier did not perform well when all the HTML tags were removed. This was due to too much information being lost. Removing the HTML left too little information remaining in the raw text introducing ambiguity. In the configuration of Obfuscation 2(i) HTML tags were not removed (HTML comments were still removed). This shows an improvement in the performance compared with the performance of Obfuscation 2.

With the Bayesian classifier a similar pattern emerges. HTML tags do offer valuable information, the results confirm this. The removal of HTML comments can also improve performance. Information lost through the removal of HTML comments does not affect the performance as greatly as that of removal of all HTML tags.

Obfuscation 3 was the replacement of obfuscated words with the original word. This improved the performance of Nearest Neighbour, showing about a 2% improvement over all three data sets. With Bayesian an increase in error is shown in two Data Sets while in the larger Data Set an improvement is shown. The poor performance of Bayesian classifier could be due, in part, to the large number of stopwords present in the top Z features. With the larger corpus size the number of candidate features increases and the selection of stopwords should be less likely. In addition its important to note the respective age of each of the data sets; data set 1

& 2 are both contain legitimate e-mail from one year ago where as data set 3 is from this current year(Autumn 03 – Summer 04).

Obfuscation 4 was to study the impact of a non-token features on the performance of classifiers. The non-token feature was a character count which considered the number of alphanumeric, white space and non-alphanumeric characters in the body of an e-mail. For each a percentage was calculated and this was used in creating the feature.

This configuration showed improvement over all Data Sets and on both classifiers. Spam normally contains excessive use of punctuation, especially if obfuscation is involved, similarly with white space. Measuring the percentage of the e-mail that consisted of punctuation or white space seems a logical development. The constructed non-token features regularly occurred in the selected features. This shows that even simple non-token features can have a positive impact on accuracy.

Obfuscation 5 was another non-token feature, it checked for the presence of obfuscation. This experiment was not a success, as the results obtained matched exactly with baseline figures. The most likely cause of this would be a fault in the implementation. This will be investigated in future work.

Finally pair tokens, which promised to offer much more information, seem to actually decrease accuracy. Over all Data Sets and on both classifiers an increase in error was noted. This increase could be due overfitting caused by the actual pair tokens produced. For example concatenation of HTML tags “<tr>^<td” while a valid feature might not offer as much information as say “buy^Viagra”. The later normally only occurs in spam while the former can occur in both spam and legitimate e-mail.

From the results obtained the following conclusions can be made. Stopword removal is worthwhile and has shown to increase accuracy. Certain types of

obfuscation removal are valuable, while others increase error dramatically. A combination of Obfuscation 1, 4 and 2(i) could help decrease error further. The exact combination is still under investigation. It is also unclear whether or not stopwords should be removed before constructing pair tokens. Certain stopwords when concatenated with other tokens can be very useful pair tokens. For example “for^FREE” or “a^pill” or “one^inch”.

The experiments conducted show that there is scope to investigate the stages prior to classification. A lot of work seems to involve developing more advanced classifiers. Development of pre-processing engines which can remove obfuscation, remove stopwords and construct non-token features are advantageous.

6.2 Other Approaches and Trends

Other technologies such as Sender Policy Framework [SPF03] offer a new approach to anti-spam. Their aim is to attach identity to e-mail; a similar thread of work is being done by Microsoft in their caller identity project [Microsoft04]. Both need widespread adoption in order to succeed. Microsoft may have enough power in order to roll out such measures but the majority of MTAs on the internet do not run Microsoft products. I do admire the SPF solution, but only as a first stage filtering process and even then I can envision problems with users configure MTAs and MUAs incorrectly. One of the only reliable methods of filtering spam is content based spam filtering. There have already been reports that spam senders are using SPF, much faster than legitimate senders have [theRegister04].

Would the world be a better place with no spam? The burden on e-mail may subside but spammers will try other forms of communication to send their advertisements. The author believes that mobile technology and devices is the next target. Since they are or are becoming ubiquitous it is only a matter of time before some person tries to exploit this.

“Just because there is a police force does not mean there is no crime” – Anon. This is particularly true in the case of spam. Anti-spam measures have been deployed for years yet spam is still a problem. Bill Gates (Microsoft) claimed he (Microsoft) will solve the problem of spam in two years, the author and many others are very sceptical. There will not be a panacea for e-mail spam, it will slowly reach a plateau and then begin to subside as new more profitable communication medium are exploited.

6.3 Future Work

This thesis is a starting place for future research. It has investigated a few theories about spam classification. A platform has been developed on future experiments can be conducted and data has been collected which can be used in future experiments.

There are numerous other avenues to investigate. The use of Part of Speech tagging in identifying spam is something that could help increase accuracy of classifiers. More advanced pair token generators have been used in other spam classification systems. Investigating and improving pair token generation could lead to better results. More advanced obfuscation removal and identification techniques are also a good research area.

Header information was not included when extracting information from spam in these experiments. Header information can reveal a lot of information. Exploiting this should lead to much better results.

SMTP proxies are a good idea and developing one which could be used to conduct experiments on is a research goal. Having real world data is a serious advantage when doing spam classification. With such a system instantaneous results could be collected.

Glossary

Deobfuscate: Removal or replacement of obfuscation introduced by spammers.

Error: This corresponds to the number of misclassifications there were on a given set of instances. If the size of the input set is N then the error is the sum of the false positives and false negatives divided by the size of the input set. See Equation 6 - Error for the mathematical formula.

False Negative (FN): A false negative is when a classifier incorrectly identifies an instance as being negative. In this case a false negative is when a spam is classified as being a legitimate e-mail.

False Positive (FP): A false positive refers to when a classifier incorrectly identifies an instance being positive. In this case a false positive is when a legitimate e-mail is classified as a spam. As one can expect this is a serious problem. Loss of information due to false positives can be catastrophic.

Mode: The mode is the value that appears most often in the sample.

MUA: Mail utility application. This is sometimes known as the mail client. It is a piece of software which allows the end-user to compose and read e-mail. Some common MUAs are Microsoft Outlook, Microsoft Outlook Express, Eudora and Mutt.

MTA: Mail Transfer Application. This is the piece of software which runs on servers. Its role is to transfer e-mail, received from other MTAs or from MUAs to its destination.

Phishing: A form of spam scam whereby the spammers send you an e-mail claiming to be from a Bank or similar. The e-mail normally asks for your account details.

Second Hand Spam: This is spam that is submitted to a corpus by individuals. These individuals will forward spam they receive to the archive. It is stored in the archive, where other individuals can use and download the spam – normally for spam filtering exercises.

Spam-Spasm: When a user accidentally deletes a legitimate e-mail when purging the inbox of spam. It normally happens where there is a lot of spam in the inbox.

Stopwords: A list of common or general terms (e.g., prepositions, and articles). These are often removed from text as the value of using these words to classify or search for documents is practically nil. This is due to the fact that they occur so frequently in documents/e-mails. The removal of stopwords is highly common in information retrieval tasks.

True Negative (TN): A true negative is where a classifier correctly identifies an instance as being negative. In this case a true negative is when the classifier correctly identifies a legitimate e-mail. Again this is ideal since we want legitimate e-mail to make it through the filtering process.

True Positive (TP): A true positive is where a classifier correctly identifies an instance as being positive. In this case a true positive is when the classifier correctly identifies a spam (i.e. given the input is a spam the classifier should identify it as a spam). Once the classifier has identified the spam we can filter them out.

Web-Beacon: Another term for a web bug. See page 12 for more details

Wild Spam: defined as spam collected by myself from active e-mail accounts. This is in contrast to Second Hand Spam which is spam that is received by others and forwarded to a corpus. I would consider SpamArchive.org as containing Second Hand Spam.

Zombie: A Zombie is a term for hosts which have been compromised by crackers/spammers which are being used for the sole purpose of relaying spam. Zombies generally have a high bandwidth connection to the internet (DSL or Cable) as this is needed when sending large numbers of spam. The owner of the zombie host generally does not know their host has been compromised. A Trojan is a common way of infecting unsuspecting hosts.

References

[Androutsopoulos03]
Androutsopoulos I, Sakkis G, Paliouras G, Karkaletsis V, Spyropoulos C D, Stampatopoulos P. (2003). A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists, Information Retrieval, 6, 49 – 73.

[Cumming04]
Dr. John Graham-Cumming (2004). Talk given at Spam Conference 2004 on How to beat an adaptive spam filter http://www.jgc.org/

[ePrivacy03]
ePrivacy Group (2003). Spam by Numbers, ePrivacy Group, a Philadelphia-based trust technology company.

[Feistel73]
Feistel, H. (1973). Cryptography and Computer Privacy. Scientific American. 228(5): 15-23

[Graham02]
Graham P. (2002) A Plan for Spam. http://paulgraham.com/spam.html

[Judge03]
Judge P (2003) The State of the Spam Problem. New Horizons, Educause review September/October.

[Microsoft04]
Microsoft Corporation (2004). Caller ID for E-Mail, The next step to Deterring Spam.

[Mitchel97]
Mitchel, T. (1997) Machine Learning, McGraw-Hill International Edition.

[Ol2mox]
Outlook to UNIX mail converter, http://sourceforge.net/projects/ol2mbox

[Quinlan86]
Quinlan J. R. (1986) Induction of decision trees. Machine Learning 1(1), 81 – 106

[Rawlinson76]
Graham Rawlinson (1976). The Significance of Letter Position in Word Recognition, PhD Thesis, Nottingham University
http://www.mrc-cbu.cam.ac.uk/personal/matt.davis/Cmabrigde/rawlinson.html

[SPF03]
Danisch h, Fecyk G, Lentczner M, Weng Wong M. (2003) Sender Permitted From, A convention to Describe Hosts Authorised to Send SMTP Traffic. Rename to Sender Policy Framework in 2004. http://spf.pobox.com/

[theRegister04]
Leyden J. (2004) Spammers embrace email authentication. http://www.theregister.co.uk/2004/09/03/email_authentication_spam/

[WebsTav85]
Webster, A. and S. Tavares. (1985). On the Design of S-Boxes. Advances in Cryptology, CRYPTO '85 523-534.