Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

Spam Classification Using Machine Learning Techniques - Sinespam

José Norte Sosa

Advisor: PhD. René Alquezar Mancho

August 2010

Content

1 - Introduction & Objectives

Most e-mail readers spend a non-trivial amount of time regularly deleting junk e-mail (spam) messages, even as an expanding volume of such e-mail occupies server storage space and consumes network bandwidth. An ongoing challenge, therefore, rests within the development and refinement of automatic classifiers that can distinguish legitimate e-mail from spam. Some published studies have examined spam detectors using Naïve Bayesian approaches and large feature sets of binary attributes that determine the existence of common keywords in spam, and many commercial applications also use Naïve Bayesian techniques.

Spammers recognize these attempts to prevent their messages and have developed tactics to circumvent these filters, but these evasive tactics are themselves patterns that human readers can often identify quickly. This work had the objectives of developing an alternative approach using a neural network (NN) classifier brained on a corpus of e-mail messages from several users. The features selection used in this work is one of the major improvements, because the feature set uses descriptive characteristics of words and messages similar to those that a human reader would use to identify spam, and the model to select the best feature set, was based on forward feature selection. Another objective in this work was to improve the spam detection near 95% of accuracy using Artificial Neural Networks; actually nobody has reached more than 89% of accuracy using ANN.

1.1 - What is "spam"

Spam, in computing terms, means something unwanted. It has normally been used to refer to unwanted email or **Usenet messages**, and it is now also being used to refer to unwanted Instant Messenger (IM) and telephone Short Message Service (SMS) messages. Spam email is unwanted, uninvited, and inevitably promotes something for sale. Often the terms junk email, Unsolicited Bulk Email (UBE), or Unsolicited Commercial Email (UCE) are used to refer to spam email. Spam generally promotes Internet – based sales, but it also occasionally promotes telephone- based or other methods of sales too.

People who specialize in sending spam are called spammers. Companies pay spammers to send emails on their behalf, and the spammers have developed a range of computerized tools and techniques to send these messages. Spammers also run their own online businesses and market them using spam email.

The term "spam email" generally precludes email from known sources, regardless of however unwanted the content is. One example of this would be an endless list of jokes sent from acquaintances. Email virus, Trojan horses, and other malware (short for malicious software) are not normally categorized as spam either, although they share some common traits with spam. Emails that are not spam are often referred to as ham, particularly in the anti-spam community. Spam is subjective, and a message considered spam by one recipient may be welcomed by another.

Anti-spam tools can be partially effective in blocking malware; however, they are best at blocking spam. Special ant-virus software can and should be used to protect your inbox from other undesirable emails.

1.2 - Definitions

The following definitions will be used throughout this work:

- Spam: Unsolicited commercial Email or UCE. It is any email that has not been requested and contains an advertisement of some kind.
- Ham: The opposite of spam – email that is wanted
- False Negative: A spam email message that was not detected successfully.
- False Positive: A ham email message that was wrongly detected as spam.

1.3 - The History of Spam

Here are some important dates in the development of the internet:

- 1969: Two computers networked via a router
- 1971: First email sent using a rudimentary system
- 1979: Usenet (newsgroups) established
- 1990: The World Wide Web concept born
- 2004: The Internet is a major global network responsible for billions of dollars of commerce.

There is one omission from this time line:

- 1978: The first email spam sent.

Spam has been part of the Internet from a relatively early stage in its development. The first spam email was sent on May 3rd, 1978, when the U.S. Government funded Arpanet, as it was called then. The first spammer was a DEC engineer called Gary Thuerk who invited recipients of his email to attend a product presentation. This email was sent using the Arpanet, and caused an immediate response from the chief of the Arpanet, Major Raymond Czahor, at the violation of the non-commercial policy of the Arpanet.

Spam really took off in 1994 when an Arizona attorney, Laurence Carter, automated the posting of messages to many internet newsgroups (Usenet) to advertise his firm´s services. The resultant outcry from Usenet users included the coining of the term "spam", when one respondent wrote "Send coconuts and cans of Spam to Cantor & Co.". This sparked the beginning of spam as it is now experienced.

Spam email has increased in volume as the Internet has developed. In April 2009, *PC Magazine* reported that 98% of all email is spam.

1.4 - Spammers

Typically, spammers are paid to advertise particular websites, products and companies, and are specialists in sending spam email. There are several well-known spammers who are responsible for a large proportion of spam and have evaded legal action.

Individual managers of websites can send their own spams, but spammers have extensive mailing list and superior tools to bypass spam filters and avoid detection. Spammers have a niche in today´s marketing industry, and their clients capitalize on this.

Most spam emails are now sent from "Trojanned" computers, as reported in a press release by broadband specialist. The owners or users of trojanned computers have been tricked into running software that allows a spammer to send spam email from the computer without the knowledge of the user. The Trojan software often exploits security holes in the operating system, browser, or email client of a user. When a malicious website is visited, the Trojan software is installed on the computer. Unknown to users, their computer may become the source of thousands of spam email a day.

1.5 - Emerging spam for social networking attacks

With individuals and businesses hooked on online social outlets, cybercriminals have taken notice and started using them for their gain. Beyond the common nuisances, such as wasted company time and bandwidth, malware and malicious data theft issues have presented serious problems to social networks and their users. Spam is now common on social networking sites, and social engineering—trying to trick users to reveal vital data, or persuading people to visit dangerous web links—is on the rise.

Social network logon credentials have become as valuable as email addresses, aiding the dissemination of social spam because these emails are more likely to be opened and trusted than standard messages. In many cases, spam and malware distribution are closely intertwined.

United States 14.4%
Brazil 11.3%
India 5.3%
China 5.1%
South Korea 4.9%
Turkey 3.8%
Poland 3.7%
Vietnam 3.7%
Russia 3.2%
Spain 2.9%
Italy 2.4%
Argentina 2.3%
Other 37%

Dirty dozen spam-relaying countries

● Asia 34.9%
● Europe 25%
● South America 19.5%
● North America 17.5%
● Africa 2.1%
● Oceania 0.6%
● Other 0.3%

Spam by continent
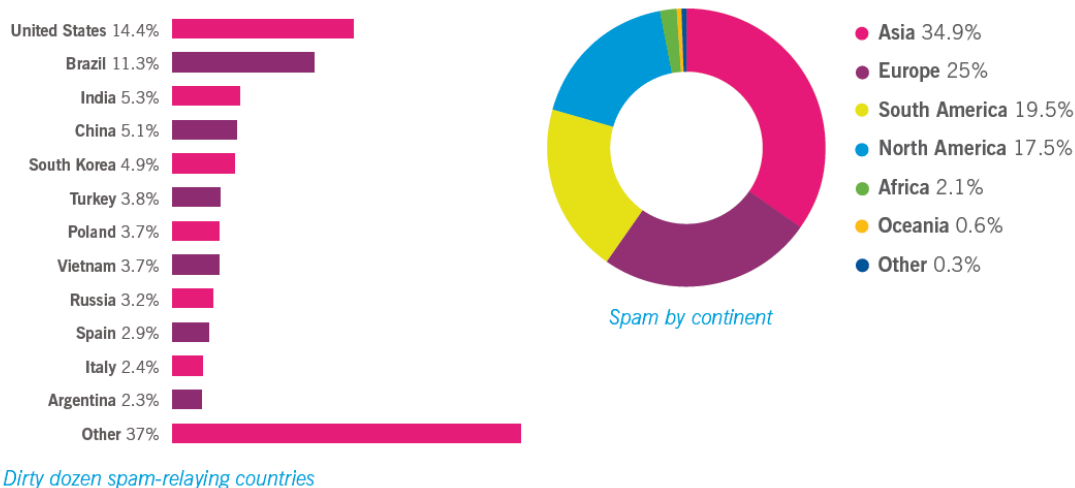
Source: Sophos Inc. – 2010 World Spam Review – Spain is part of the top 10

1.6 - The Costs of Spam

Spam is very cheap to send, the cost are insignificant as compared to conventional marketing techniques, so marketing by spam is very cost-effective, despite very low rates of purchases in response. But it translates into major costs for the victim.

1.7 - Costs to the Spammer

A report by Tom Galler, Executive Director of SpamCon Foundation, estimated that the cost to send a single spam email was as little as one thousandth of a cent, yet the cost to the recipient was around 10 cents.

The overhands in sending spam are low. The main costs are:

- **An Internet connection**: there are lots of flat-rate Internet Service Providers (ISPs) offering packages at around €20/month. A spammer doesn´t particularly need a Digital Subscriber Line (DSL) or cable modem service, a dial-up connection will also allow large quantities of spam to be sent. In fact, dial-up accounts are preferable, as spammer accounts are routinely shut down when complaints about spam are received. Dial-up accounts are easy to set up and can quickly activate within minutes, but DSL typically has a lead time of days.
- **Software**: specialist spam software is essential. A normal email client will restrict the number of spam messages that can be sent, and require the spammer to spend more time in front of the computer. Spammers usually write their own software, steal someone else´s, or buy one. A spammer with some technical knowledge and starting from scratch can have software ready after a week. To pay someone to develop that software would cost the spammer € 1000.
- **A mailing list:** most spammers will build up their own list of email addresses. For beginners, it is possible to buy a CD with 6 million email addresses on it for around €50. Ironically, these CDs are marketed via spam email. Email addresses that are guaranteed to be currently active sell for larger sums.
- **A web server:** this is an optional cost. It allows a spammer to deliver "web bug" images to validate their mailing list. Web bugs are discussed further in later chapters. Basic web hosting packages cost less than € 10 a month.

For less than € 1100, plus monthly cost of less than € 160, a spammer could have the software, internet connection, and a supply of addresses required to be operational.

A single computer can send thousands of emails an hour using dial-up. Spam varies, but a typical message size might be around 6,000 bytes. On a fast dial-up of around 50Kps, it would take one second to send this email to one recipient. It would take only a little longer to send it to 100 recipients. In other words, at least 3,600 emails can be sent in an hour. For smaller emails, the number sent per hour would be greater. The spammer needs to invest 15 minutes of their time, and the software will continue to send spam for many hours. With three phone lines, they could work for a total of an hour, and send approximately 10,000 emails an hour or 200,000 a day or more using DSL line.

1.8 - Costs to the Recipient

The European Union performed a study into UCE in 2009. In the findings, it estimated the cost of receiving spam borne consumers and businesses to be around € 8 billon. These costs are partly incurred through lost productivity or time, partly in direct costs, and partly in direct costs incurred by suppliers, and passed on.

The cost of spam in a commercial environment is estimated to be as high as € 600 to € 1000 per year, per employee. For a 50-person company, this cost could be as high as € 50.000 per year. Spam emails distract or take employees time and use disk space, processing power, and network bandwidth. Removing spam by hand is time consuming and laborious when there is a large amount of spam. In addition, there is a business risk, as genuine messages may be removed along with unwanted ones. Spam can also contain unsavory topics that some employee's won´t tolerate.

1.9 - Spam and the Law

In the USA, legislation proceeding on spam has been in progress since 1997. The latest legislation is the CAN-SPAM act (Bill number S.877) of 2009. This supersedes many state laws and is currently being used to prosecute persistent spammers. However, it is not proving a deterrent; the **Coalition Against Unsolicited Commercial Email** (CAUCE) reported in June 2008 that despite several high-profile lawsuits by the Federal Trade Commission (FTC) and ISPs, spam volumes were still increasing. The CAN-SPAM act is seen as weak on two counts: that consumers have to explicitly opt-out from commercial emails, and secondly, only IPSs can take action against spammers.

In Europe, legislation exists that makes spamming illegal. However, when Directive 2002/58/EC was passed in 2002, there were several problems with it. Business-to-business emails were excluded – a business could spam each and every account at any other business and stay within the law. Additionally, each individual member state has to pass its own laws and penalties for offenders. The law requires spammers to use opt-in emailing, where recipients have to explicitly request to receive commercial emails rather than the opt-out model proposed in the USA, where anyone can receive spam and has to request to be removed from mailing lists.

The Guardian, a UK newspaper, reported in June 2009 that gangs of spammers are moving their operations to the UK due to the leniency of the laws there. The maximum penalty they face in the UK is 5000 (pounds), while in Italy spammers face up to three years of imprisonment. Until June 2004, no one had been convicted under this act in the UK.

In Australia, the spam Act 2003 came into effect in April 2004. This makes spam illegal, using an opt-in model. Additionally, there have already been successful prosecutions for spamming in Australia using previous laws.

The internet is a multinational network and domestic legislation cannot reach to another country. A U.S-based spammer would be at risk of prosecution if it spammed U.S. citizens and advertised a product made and sold in the US. But the spammer from the Far East would be at very little risk of prosecution. Domestic legislation will not affect the volume of spam, but it may occasionally affect the types of products advertised via spam.

Spammers will often reroute spam via other nations, so spam is sent from the US to another country and relayed back to the US again. This makes it more difficult to trace the source of the email and to prosecute them. Many countries have no anti-spam laws and there is little or even no risk to the spammer. The blurring of geographical boundaries by the Internet does little to aid in tracing spam email to its source. Anti - spam in now moving towards tracking

spammers through other means. In May 2008, the New2008, the New York Times reported that the Direct Marketing Association is using paper trails in the real world to trace spammers in the virtual world with success.

2 - Spam Techniques

As spam increased in volume and became more of a problem, anti-spam techniques were developed to counteract it. Tools to block spam were developed by a group of professionals. These tools were not always automated, but when used by system administrators of large sites, they could successfully filter spam for a large number of users. In response, spammers evolved their techniques to increase the number of spam delivered by working around and through the filters. As spam filters improved, spammers designed other methods of bypassing the filters and the cycle repeated. This resulted in the development of both spam and anti-spam techniques and tools over a number of years. This evolutionary process continues today.

Anti-spam tools use a wide range of techniques to reduce the volume of spam received by a user. A number of these techniques will be described in following section. There are several antispam techniques based on Open Source tool that we will examine in the light of the various techniques it uses to filter spam.

2.1 - Spamming Techniques
Spammers have developed a complex arsenal of techniques for spamming. Important spamming techniques are described in the following points.

2.2 - Open Relay Exploitation
An open relay is a computer that allows *any* user to send email. Spammers use such computers to send spam without the email being trace to its true origin.

2.3 - Collecting Email Addresses
Early spammers had to collect email addresses in order to send spam. They use a variety of methods, from collecting email addresses from the Internet and Internet newsgroups to simply guessing email addresses.

2.4 - Hiding Content
Most people can detect spam from the email subject or sender. It is often easy to discard spam emails without even looking at the body. One technique used by spammers is to hide the true content of their emails. Often, the subject of an email is a simple "Hi"; alternatively, an email might appear to be a reply to a previous email, for example "Re: tonight". Other tricks that spammers use includes using random name look important, for example, by alluding to a credit card or loan missed payment or work-related subject.

As spam filters block obvious spam words, such as "Viagra", spammers deliberately include misspelled words that are less likely to be filtered out; for example, "Viagra" might become

"V1agra" or "V-iaggr@". Although the human mind can easily translate the meaning of misspelling unconsciously, a computer program will not associate these words with spam.

2.5 - Statistical Filter Poisoning
Statistical filter poisoning involves including many random words within an email to confuse a statistical filter. *Statistical filters are described in the Anti-Spam Techniques section.*

2.6 - Unique Email Generation
To combat email content databases, which store the content of known spam emails doing the rounds, spammers generate unique emails. To confuse the email content database, the spammer only needs to change one random word in the main body of the email. One popular technique is to use the recipient´s name within the body of an email.

2.7 - Trojanned Machines
Spammers are limited by the speed of their internet connection, be it DSL or dial-up. They are also directly traceable through ISP records. A recent trend among spammers is to use PC virus technology to infect innocent users 'computers with virus-like programs. These programs send spam from the innocent parties` PCs. Such an infection is commonly called a Trojan, after the story of the Greeks invading the city of Troy by surreptitious means.

3 - Anti-Spam Techniques

As the techniques to deliver spam have become more sophisticated, so have the techniques to detect and filter spam from legitimate email. The main techniques are described in the following points. These techniques can be used on the email server by a system administrator, or an anti-spam service can be purchased from an external vendor.

3.1 - Keyword Filters
Filters are based upon common words or phrases in an email body, for example "buy", "last chance", and "Viagra". Some open source software includes a variety of keyword filters and allows easy addition of new rules.

3.2 - Open relay Blacklist
Open relay blacklists (ORBLs) are lists of open relays that have been reported and added to these blacklists after being tested. Anti-spam tools can query open relay blacklists and filter out emails originating from these sources. Some open source software can integrate with several open relay blacklist.

3.3 - ISP Complaints
It has always been possible to complain to an ISP about a spammer. Some ISPs take complaints seriously, give a single warning, and after another complaint, they terminate the account of

the offender. Other ISPs take a less active approach to spam that will rarely stop a spammer. Spammers naturally gravitate towards ISPs that are lenient with spammers.

ISP complaints remain a manually managed techniques, due to the effort thet might be wasted if an automatic report is wrong and email reported is not spam. The website http://www.spamcop.net can examine an email; determine where ISP report should be directed, and send appropriate messages of complaint to the corresponding ISPs.

3.4 - Statistical Filters

Statistical filters are those that learn common words in both spam *and* ham. Subsequently the data collected is used to examine emails and determine whether they are spam or ham. These filters are often based on the mathematical theory called Bayesian analysis. Statistical filters need to be trained by passing both ham and spam emails through, enabling the filter to learn the difference between the two. Ideally, a statistical filter should be trained regularly, and some anti-spam tools allow statistical filters to be trained automatically.

3.5 - Email Header Analysis

The software that spammers use often generates unusual headers in the emails produced. Anti-spam tools can detect these unusual headers and use them to separate spam from ham. Some open source software includes many email header test.

3.6 - Non-Spam Content Test

There are possibilities that ham emails could inadvertently trigger some anti-spam tests. For example, many emails are legitimately but unfortunately routed though a blacklisted open relay. Non-spam content test indicate that an email is not spam. They are usually created specifically for an individual or organization.

Non-spam content test are rarely shared in public, as they are specific to an industry or company, and should not get into the hands of spammers as they would start using this information to their advantage. Some open source software, allows user to created rules that will subtract from the score of an email if certain content is received. An email administrator might add negative rules for the names of products sold by the company or for industry-related jargon.

3.7 - Whitelists

Whitelists are the opposite of blacklists – lists of emails senders who are trusted to send ham and not spam. Emails from someone listed on a whitelist will normally not be marked as spam, no matter what the content of their email.

3.8 - Email Content Databases

Email content databases store the content of spam emails. These work because the same spam email will often be sent to hundreds or thousands of recipients. Email content databases store these emails and compare the content of new emails to that contained in the database. A single person reporting a spam email to such a databases will assist all other users of the service. Some open source software can integrate with several email content databases automatically.

3.9 - Sender Validation systems

A slightly different approach to spam is taken by sender validation systems. In these systems, when an email is received from an unknown source, the source is sent a challenge email. If a valid response is received to such an email, then the sender is added to a whitelist, the original email is delivered to the recipient, and the sender is never sent a challenge again.

This is effective as generally spammers use forged sender and reply-to addresses and do not received replies to the spam they send out. Consequently, the challenge is never received. In addition, spammers do not have the time to validation requests.

Some systems cleverly integrate with the user`s outgoing email an addresses book to automatically add known contacts to a whitelist. Sender validation systems are proprietary and may involve annual licensing costs or large initial fees.

Sender validation systems are inconvenient when subscribing to mailing list. Few email list administrators will respond to challenge, so the user might end up not receiving emails from the list. With most systems, it is possible to manually add addresses to the whitelist to avoid a challenge or response being required, but in the case of mailing lists, the addresses that emails are sent from may not be known until emails are received. Some open source software does include sender validation features.

3.10 - Sender Policy Framework (SPF)

The Sender Policy Framework (SPF) can be used to ensure that an email is from a valid source. It validates that a user sending email from a particular email address is permitted to send email from their current machine. SPF is a recent development and is being introduced relatively quickly. It uses additional Domain Name System (DNS) record to state which machine can send email for a domain. Some open source software uses the current draft standards for SPF.

4 - Detecting Spam

4.1 - Contents Tests

Contents tests analyze the message part of the email, and sometimes the headers. These tests typically look for key words or phrases within emails. Usually, when using content tests, a scoring system is used. It is not uncommon for words normally associated with spam emails to also appear in legitimate emails, so a score or count of suspicious words is accumulated for each email. Each word associated with spam increases the overall score of an email. The final score is compared with a predefined threshold; this is used to decide whether an email is spam or ham.

Content tests need not focus on single words; phrases and sequences of punctuation are used. The words, phrases, and other symbols tested are normally generated by a developer, who analyzes spam and manually creates tests.

Sometimes the message headers are examined as part of a content test. The message heads include dates, time, and other attributes, such as the mail application used. Often, spam-creation programs contain errors or misspellings in their headers that can be caught by spam filters.

Spammers attempt to avoid detection by deliberate misspelling and varying content slightly in each spam or spam run.

A simple example of a content test would be to locate the word "Viagra" within an email. A more complex content test would be to locate the sequence of characters `v?i?a?g?r?a´, where the `?´ represents any character, and one or more instances of `?´may not be present at all. For example, `VIAGRA´, `V I A G R A ´, and `V*i*agra´ would all match.

4.2 - Header Tests

Header tests focus on the message headers. The tests are mainly concerned with detecting fake headers and determining whether a message has been routed via an open relay.

For example, a header test could flag all emails that appear to have been sent over 72 hours ago, or sent at a future date. Most emails servers have accurate clocks. However, spammers frequently use trojanned PCs, which may have inaccurate clocks, and so spam messages might have dates that are in the past or the future. Examining email headers is described in more detail later in this section.

These tests use up considerable amounts of CPU, Memory, and disk I/O resources.

4.3 - DNS – Based Blacklists

There are many DNS – based blacklists (DNSBLs). These are also known simply as **blacklists** or **blocklists.** They provide a service that is used by MTAs (**M**essage **T**ransfer **A**gent or **M**ail

Transfer **A**gent, the store and forward part of a messaging system like email) and spam filters to indicate sites that are related to spammers. An MTA or spam filter may use one or more blacklists. Some open source software can use blacklists to filter spam.

Blacklists can generally be placed in one or more of these categories.

- A list of known open relay
- A list of known sources of spam
- A list of sites hosted by an ISP that encourages spammers in some way

Every blacklist has unique policies for adding and removing domains from the list. Some are very aggressive, and block not only sources of spam, but also any address served by the same Internet Service Provider. The intention of this approach is to force ISPs to stop doing business with spammers and thus force them off the net. This approach, called the *Internet black hole of death* has been used with success against major ISPs in the past.

Blacklists provide a spam filter or MTA with the ability to query the blacklist to see if a particular IP address is listed. If the IP address is listed, then incoming email from that host is often rejected.

Generally, IP addresses are listed on a blacklist only if have been reported. Reporting is either done by a human after examining the headers of an email message, or by an automatic system. Some blacklists remove addresses from their list after a period of time, while some will wait for proof that spam problem has stopped.

Some blacklists will test a site to see if it is running an open relay. The tests are usually automatic, probing port 25 in a similar way as the manual test.

It is the responsibility of system administrators and end users to get spammer´s machines listed on blacklist. Some spam filter software submits suspected addresses to blacklists automatically, which is a dangerous approach. Automatic systems can get out of control if unforeseen circumstances occur and a open relay backlist (ORBL) could get flooded by false reports. It is better to provide the user with an option to report a relay to a blacklist than do it automatically. This relies on having software developers provide an option for users, and the users having enough knowledge to determine whether to submit the request.

 ORBL´s generally use network I/O rather than CPU, and disk I/O. The blacklist test use lesser network I/O than is used when receiving an email. These test are suited to parallel processing systems (processing many emails at once) as the results take time to be retrieved and the machine is free to use other resources((CPU, memory, and disck I/O) for other test.

4.4 - Statistical Test

Various statistical techniques can be used to identify spam. These generally involve a training phase, where a database of spam and ham emails is taught to the filter or passed through it to identify typical characteristics of spam and ham. This allows future emails to be identified based on the learning from past emails. The various statistical techniques vary in their choice of tokens and the algorithms they use to predict whether an email is spam or ham. The tokens

used are normally words, but can include emails headers, HTML markup within emails, and others characters such as punctuation marks.

Statistical filters rely on regular training. They use the knowledge gained in training to estimate the probability that new emails are spam. As spam change, the filter must adapt in order to continue to detect the spam.

Statistical tests are resource intensive, using CPU, memory, and disk I/O.

4.5 - Message Recognition

Often, a spammer will send exactly the same message to many recipients. Although message headers may be different in each email, an email with the same body may be sent to many recipients. This has led to the creation of several anti-spam networks that contain a database of spam emails. By comparing incoming emails with the contents of this database, it is possible to quickly filter out known spam message.

To avoid sending the whole email across the network and comparing each character of line, a hash value is calculated and used. Hashing is a mathematical process that creates a small signature from a larger message. It is very unlike that two messages will have the same hash value, and so comparing hashes values is statistically the same as comparing the whole message. As the hashes values are much shorter than an email message, comparing hashes values is significantly quicker than comparing the whole message.

The calculation of a hash value is a CPU-intensive task, and there is some network I/O and related latency involved while querying the database. This test is suited to parallel processing.

5 - Artificial Neural Network Overview

An artificial neural network is a collection of connected neurons. Taken one at a time each neuron is a rather simple. As a collection however, a group of neurons is capable of producing complex results. In the following sections we will briefly summarize mathematical models of a neuron, neuron layer and neural network before discussing the types of behavior achievable from a neural network.

5.1 - Artificial Neuron Modeling

A model of a neuron has three basic parts: input weights, a summer or aggregation function, and an output function. The input weights scale values used as inputs to the neuron, the summer adds all the scaled values together, and the output function produces the final output of the neuron. Often, one additional input, known as the bias is added to the model. If a bias is used, it can be represented by a weight with a constant input of one. This description is laid out visually below.
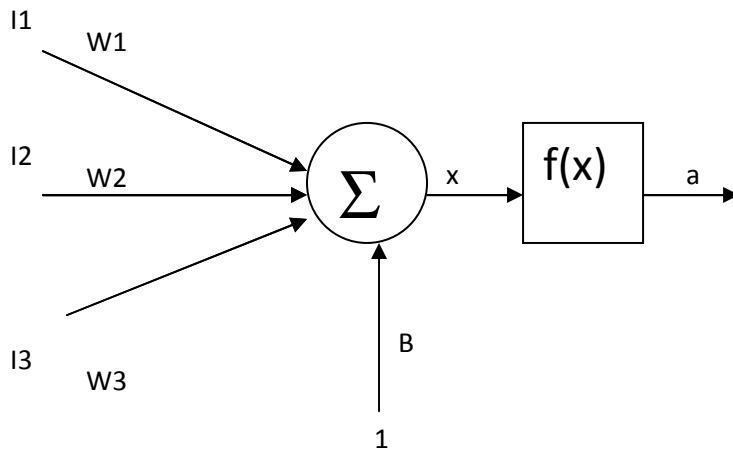
where **I1**, **I2**, and **I3** are the inputs, **W1**, **W2**, and **W3** are the weights, **B** is the bias, **x** is an intermediate output, and **a** is final output. The equation for **a** is given by $a = f(W_1 I_1 + W_2 I_2 + W_3 I_3 + B)$ where f could be any function. Most often, f is the sign of the argument (i.e. 1 if the argument is positive and -1 if the argument is negative), linear (i.e. the output is simply the input times some constant factor), or some complex curve used in function matching e.g. Logistic or hyperbolic tangent.

Logistic (Sigmoid logistic) - We have found this function useful for most neural network applications. It maps values into the (0, 1) range. Always use this function when the outputs are categories.
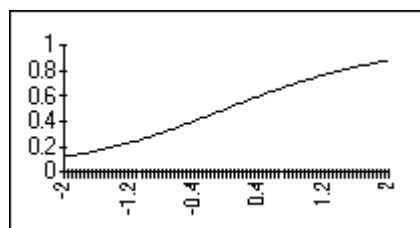
Logistic Function

Linear - Use of this function should generally be limited to the output slab. It is useful for problems where the output is a continuous variable[Jordan, 1995], as opposed to several outputs which represent categories. Stick to the logistic for categories. Although the linear

function detracts from the power of the network somewhat, it sometimes prevents the network from producing outputs with more error near the min or max of the output scale. In other words the results may be more consistent throughout the scale. If you use it, stick to smaller learning rates, momentums, and initial weight sizes. Otherwise, the network may produce larger and larger errors and weights and hence not ever lower the error. The linear activation function is often ineffective for the same reason if there are a large number of connections coming to the output layer because the total weight sum generated will be high.
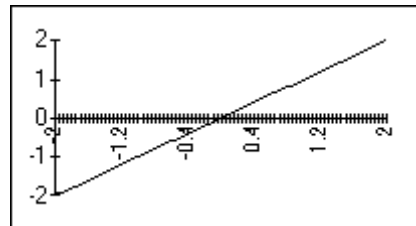
Linear Function

Tanh (hyperbolic tangent) - Many experts feel this function should be used almost exclusively. It is sometimes better for continuous valued outputs, however, especially if the linear function is used on the output layer. If you use it in the first hidden layer, scale your inputs into [-1, 1] instead of [0,1]. We have experienced good results [McCullagh, P. and Nelder, J.A. (1989)] when using the hyperbolic tangent in the hidden layer of a 3 layer network, and using the logistic or the linear function on the output layer.
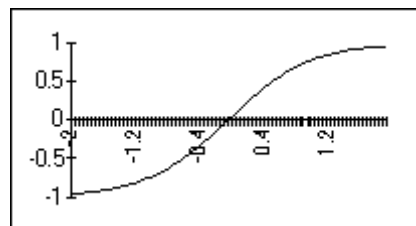
Tanh Function

When artificial neurons are implemented, vectors are commonly used to represent the inputs and the weights so the first of two brief reviews of linear algebra is appropriate here. The dot product of two vectors $\vec{x} = (x_1, x_2, \cdots, x_n)$ and $\vec{y} = (y_1, y_2, \cdots, y_n)$ is given by $\vec{x} \bullet \vec{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$. Using this notation the output is simplified to $a = f(\vec{W} \bullet \vec{I} + B)$ where all the inputs are contained in $\vec{I}$ and all the weights are contained in $\vec{W}$.

5.2 - Neuron Layer

In a neuron layer each input is tied to every neuron and each neuron produces its own output. This can be represented mathematically by the following series of equations:

$$a_1 = f_1(\vec{W}_1 \bullet \vec{I} + B_1)$$

$$a_2 = f_2(\vec{W}_2 \bullet \vec{I} + B_2)$$

$$a_3 = f_3(\vec{W}_3 \bullet \vec{I} + B_3)$$

$$\ldots$$

NOTE: In general these functions may be different.

And we will take our second digression into linear algebra. We need to recall that to perform the operation of matrix multiplication you take each column of the second matrix and perform the dot product operation with each row of the first matrix to produce each element in the result. For example the dot product of the ith column of the second matrix and the jth row of the first matrix results in the (j,i) element of the result. If the second matrix is only one column, then the result is also one column.

Keeping matrix multiplication in mind, we append the weights so that each row of a matrix represents the weights of an neuron. Now, representing the input vector and the biases as one column matrices, we can simplify the above notation to:

$$\vec{a} = f(W \cdot I + B)$$

which is the final form of the mathematical representation of one layer of artificial neurons.

5.3 – Multilayer Feed-Forward Neural Network

A multilayer feed-forward neural network is simply a collection of neuron layers where the output of each previous layer becomes the input to the next layer. So, for example, the inputs to layer two are the outputs of layer one. In this exercise we are keeping it relatively simple by not having feedback (i.e. output from layer n being input for some previous layer). To

mathematically represent the neural network we only have to chain together the equations. The final equation for a three layer network is given by:

$$a = f(W_3 \cdot f(W_2 \cdot f(W_1 \cdot \vec{I} + \vec{B}_1) + \vec{B}_2) + \vec{B}_3)$$

5.4 - Neural Network Behavior

Although transistor now switch in as little as 0.000000000001 seconds and biological neurons take about .001 seconds to respond we have not been able to approach the complexity or the overall speed of the brain because of, in part, the large number (approximately 100,000,000,000) neurons that are highly connected (approximately 10,000 connections per neuron). Although not as advanced as biologic brains, artificial neural networks perform many important functions in a wide range of applications including sensing, control, pattern recognition, and categorization. Generally, networks (including our brains) are trained to achieve a desired result. The training mechanisms and rules are beyond the scope of this work, however it is worth mentioning that generally good behavior is rewarded while bad behavior is punished. That is to say that when a network performs well it is modified only slightly (if at all) and when it performs poorly larger modifications are made. As a final thought on neural network behavior, it is worth noting that if the output functions of the neurons are all linear functions, the network is reducible to a one layer network. In other words, to have a useful network of more than one layer we must use a function like the sigmoid (an s shaped curve), a linear function, or any other non-linear shaped curve.

5.5 - Classification Problem – Training Networks

This section introduces the concept of training a network to perform a given task. Some of the ideas discussed here will have general applicability, but most of the time refer to the specific of TLUs (Threshold Logical Unit) to perform a given classification the network must implement the desired decision surface. Since this is determined by the weight vector and threshold, it is necessary to adjust these to bring about the required functionality. In general terms, adjusting the weights and thresholds in a network is usually done via an iterative process of repeated presentation of examples of the required task. At each presentation, small changes are made to weights and thresholds to bring them more in line with their desired value. This process is known as *training* the net, and the set of examples as the training set. From the network´s viewpoint it undergoes a process of learning, or adapting to, the training set, and the prescription for how to change the weights at each step is the learning rule. In one type of training the net is presented with a set of input patterns or vectors $\{x_i\}$ and, for each one, a corresponding desired output vector or target $\{t_i\}$. Thus, the net is supposed to respond with t_k, given input x_k for every k. This process is referred to as supervised training (or learning) because the network is told or supervised at each step as to what it is expected to do.

Next there is a description, of how to train the TLUs and a related node, the perceptron, using supervised learning. This will consider a single node in isolation at first so that training set

consists of a set of pairs {v, t}, where v is an input vector and t is the target class or output ("1" or "0") that belongs to.

The first step toward understanding neural nets is to abstract from the biological neuron, and to focus on its character as a threshold logic unit (TLU). A TLU is an object that inputs an array of weighted quantities, sums them, and if this sum meets or surpasses some threshold, outputs a quantity. Let's label these features. First, there are the inputs and their weights: X1,X2, ..., Xn and W1, W2, ...,Wn. Then, there are the Xi*Wi that are summed, which yields the activation level a, in other words:

 a = (X1 * W1)+(X2 * W2)+...+(Xi * Wi)+...+ (Xn * Wn)

The threshold is called theta. Lastly, there is the output: y. When a >= theta, y=1, else y=0. Notice that the output doesn't need to be discontinuous, since it could also be determined by a squashing function, s (or sigma), whose argument is a, and whose value is between 0 and 1. Then, y=s(a).
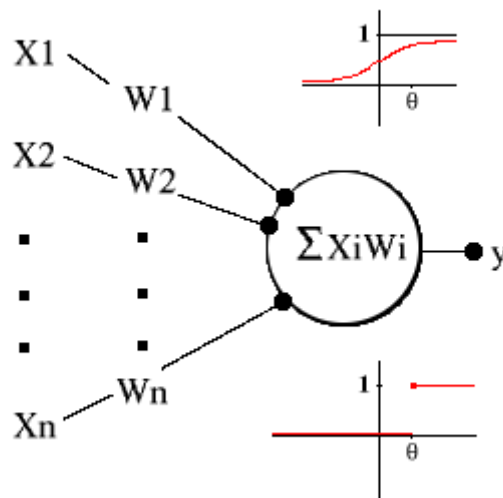
Figure 5.5.A Threshold logic unit, with sigma function (top) and cutoff function (bottom)

5.6 - Training the threshold as a weight

In order to place a adaptation of the threshold on the same footing as the weights, there is a mathematical trick we can play to make it look like weight. Thus, we normally write w·x ≥ Θ as the condition for output of a "1". Subtracting Θ from both sides gives w·x − Θ ≥ 0 and making the minus sign explicit results in the form w·x + (-1) Θ ≥ 0. Therefore, we may think of the threshold as an extra weight that is driven by an input constantly tied to the value -1. This leads to the negative of the threshold begin referred to sometimes as the bias. The weight vector, which was initially of dimension n for an n-input unit, now becomes the (n+1)-dimensional vector w_1, w_2, …… w_n, Θ. We shall call this the augmented weight vector, in contexts where confusion might arise, although this terminology is by no means standard. Then for all TLUs we may express the node function as follows:

$$w \cdot x \geq 0 \qquad \rightarrow \qquad y = 1$$

$$w \cdot x < 0 \qquad \rightarrow \qquad y = 0$$

Putting w·x = 0 now defines the decision hyperplane, which, is orthogonal to the(augmented) weight vector. The zero-threshold condition in the augmented space means that the hyperplane passes through the origin, since this is only way that allows w·x = 0. We now illustrate how this modification of pattern space works with an example in 2D.

Example:

Consider two-input TLU that ouputs a "1" with input (1,1) and a "0" for all other binary inputs so that a suitable (non-augmented) weight vector is (1/2, 1/2) with threshold 3 / 4. This is shown next in figure 5.6.A where the decision line and weight vector are displayed. That the decision line goes through the points x_1 = (1/2,1) and x_2 =(1, 1/2) maybe easily verified since according to w · x_1 = w · x_2 = 3/4 = Θ. For the augmented pattern space we have to go 3D as shown in next figure 5.6.B.
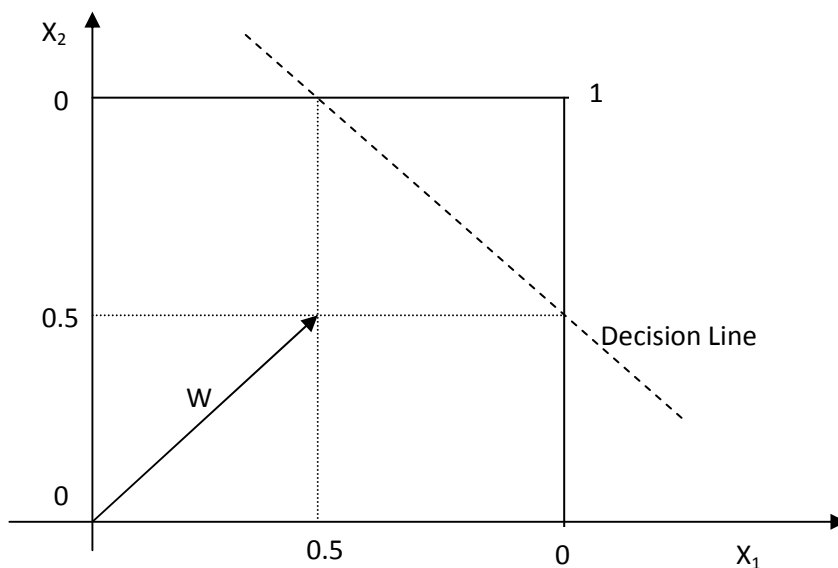
Figure 5.6.A – Two-dimensional TLU example

The previous two components x_1, x_2 are now drawn in the horizontal plane while third component x_3 has been introduced, which is shown as the vertical axis. All the patterns to the TLU now have the form $(x_1, x_2, -1)$ since the third input is tied to the constant value of -1. The augmented weight vector now has a third component equal to the threshold and is perpendicular to a decision plane that passes through the origin. The old decision line 2D is formed by the intersection of the decision plane and the plane containing the patterns.
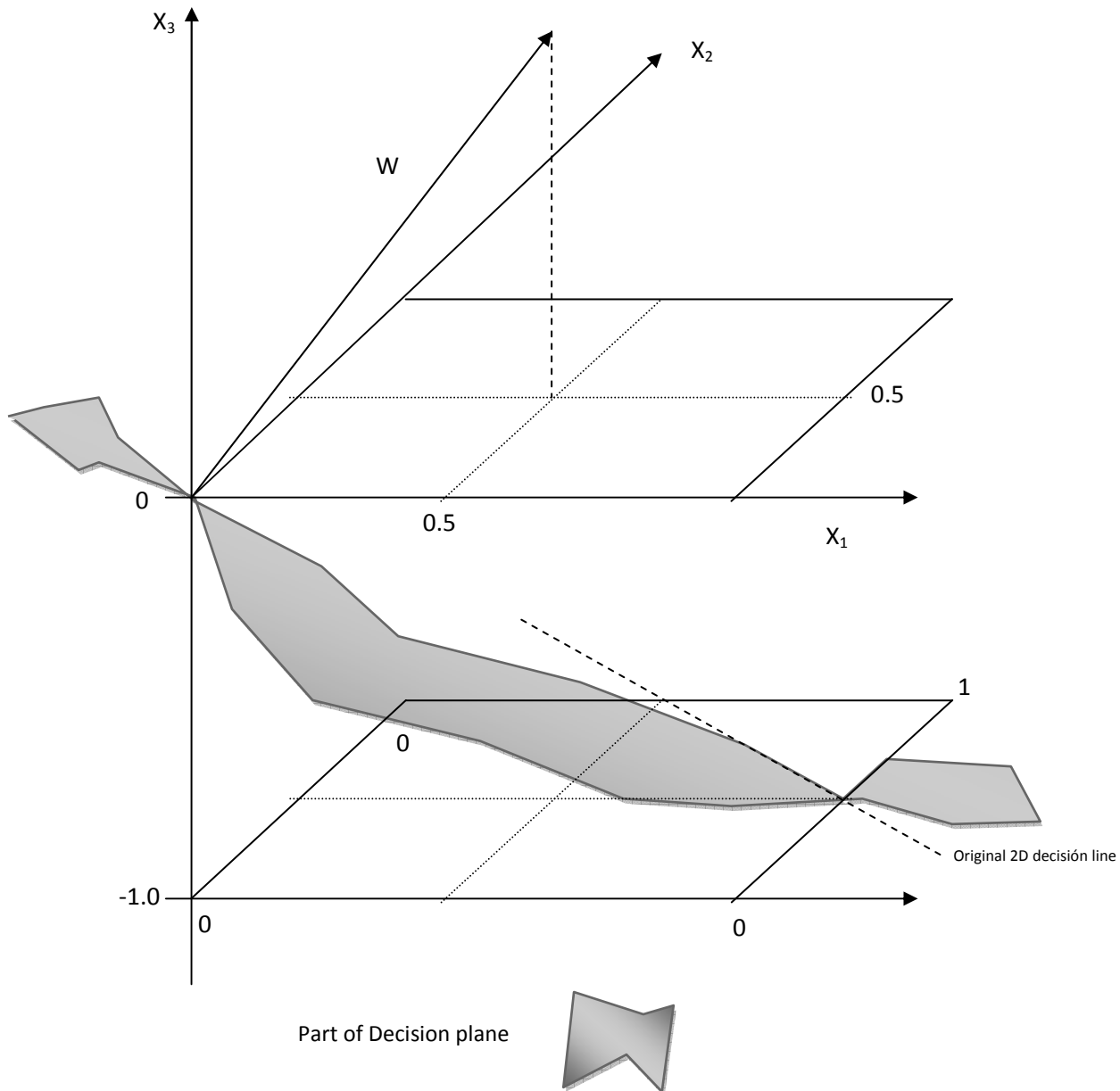
Figure 5.6.B – Two-dimensional example in augmented pattern space

5.7 - Adjusting the weight vector

We now suppose we are to train a single TLU with augmented weight vector w using the training set consisting of pairs like v, t. The TLU may have any number of inputs but we will represent that happening in pattern space in a schematic way using cartoon diagrams in 2D.

Suppose we present an input vector v to the TLU with desired response or target t = 1 and, with the current weight vector, it produces an output of y=0. The TLU has misclassified and we must make some adjustment to the weights. To produce a "0" the activation must have been negative when it should have been positive. Thus, the dot product w · v was negative and the two vectors were pointing away from each other as shown on the left hand side of figure 5.7.A.

In order to correct the situation we need to rotate w so that it points more in the direction of v. At the same time, we don't want to make too drastic a change as this might upset previous learning. We can achieve both goals by adding a fraction of v to w to produce a new weight vector w´, that is

$$w´ = w + \alpha v$$

where $0 < \alpha < 1$, which is shown schematically on the right hand side of figure 5.7.B.

Figure 5.7.A – TLU misclassification 1-0

Suppose now, instead, that misclassification takes place with the target t = 0 but y = 1. This means the activation was positive when it should have been negative as shown on the left in figure 5.7.B. we now need to rotate w away from v, which may be effected by subtracting a fraction of v from w, that is

$$w´ = w - \alpha v$$

as indicated on the left of figure 5.7.B.

We can combine as a single rule in the following way

$$w´ = w + \alpha(t-y)v$$

This may be written in terms of the change in the weight vector $\Delta w = w´ - w$ as follows:

$$\Delta w = \alpha(t-y)v$$

or in terms of components

$$\Delta w_i = \alpha(t-y)v_i \qquad : \qquad i=1 \text{ to } n+1$$
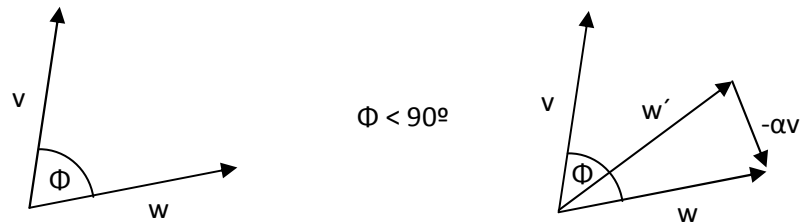
Figure 5.7.B – TLU misclassification 0-1

Where $w_n+1 = \Theta$ and $v_n+1 = -1$ always. The parameter α is called the learning rate because it governs how big the changes to the weights are and, hence, how fast learning takes place. All the forms are equivalent and define the perceptron training rule. It is called this rather than the TLU rule because, historically, it was first used with a modification of the TLU known as the perceptron. The learning rule can be incorporated into an overall scheme of iterative training as follows.

Initialize w;
Repeat
       For each training vector pair (v,t)
              Evaluate the output y when v is input to the TLU
              If y ≠ t then
                      Form a new weight vector $w'=w+\alpha(t-y)v$
              Else
                      Do nothing
              End if
       End for
Until y = t for all vectors

Algorithm 5.7 – Perceptron Learning

The procedure above entirely constitutes the perceptron learning algorithm. There is one important assumption here that has not, as yet, been made explicit: the algorithm will generate a valid weight vector for the problem in hand, If one exist. Indeed, it can be shown that this is the case and its statement constitutes the perceptron convergence theorem:

"If two classes of vector X,Y are linearly separable, then application of the perceptron training algorithm will eventually result in a weight vector w_0 such that w_0 define a TLU whose decision hyperplane separates X and Y". **Rosenblatt (1962)**

Since the algorithm specifies that we make no change to w if it correctly classifies its input, the convergence theorem also implies that, once w_0 has been found, it remains stable and no further changes are made to the weights. The convergence theorem was first proved by Rosenblatt (1962), while more recent versions may be found in Haykin (1994) and Minsky & Papert (1969).

One final point concerns the uniqueness of the solution. Suppose w_0 is a valid solution to the problem so that $w_0 \cdot x = 0$ defines a solution hyperplane. Multiplying both sides of this by a constant k preserves the equality and therefore defines the same hyperplane. We may absorb k into the weight vector so that, letting $w'_0 = kw_0 \cdot x = 0$. Thus, if w_0 is a solution, then so too is kw_0 for any k and this entire family of vectors defines the same solution hyperplane.

5.8 - The Perceptron

This is an enhancement of the TLU introduced by Rosenblatt (Rosenblatt 1962) and show in next Figure 5.8.A. It consists of a TLU whose inputs come from a set of preprocessing association units or simple A-units. The input pattern is supposed to be Boolean, that is a set of 1s and 0s, and the A-units can be assigned any arbitrary Boolean functionality but are fixed – they don't learn. The depiction of the input pattern as a grid carries the suggestion that the input may be derived from visual image for example. The rest of the node functions are just like TLU and may therefore be trained in exactly the same way. The TLU may be thought of as a special case of the perceptron with a trivial set of A-units, each consisting of a single direct connection to one of the inputs. Indeed, sometimes the term "perceptron" is used to mean what we have defined as a TLU. However, whereas a perceptron always performs a linear separation with respect to the output of its A-units, its function of the input space may not be linear separable if the A-units are non-trivial.

input pattern   Association units   weights   threshold
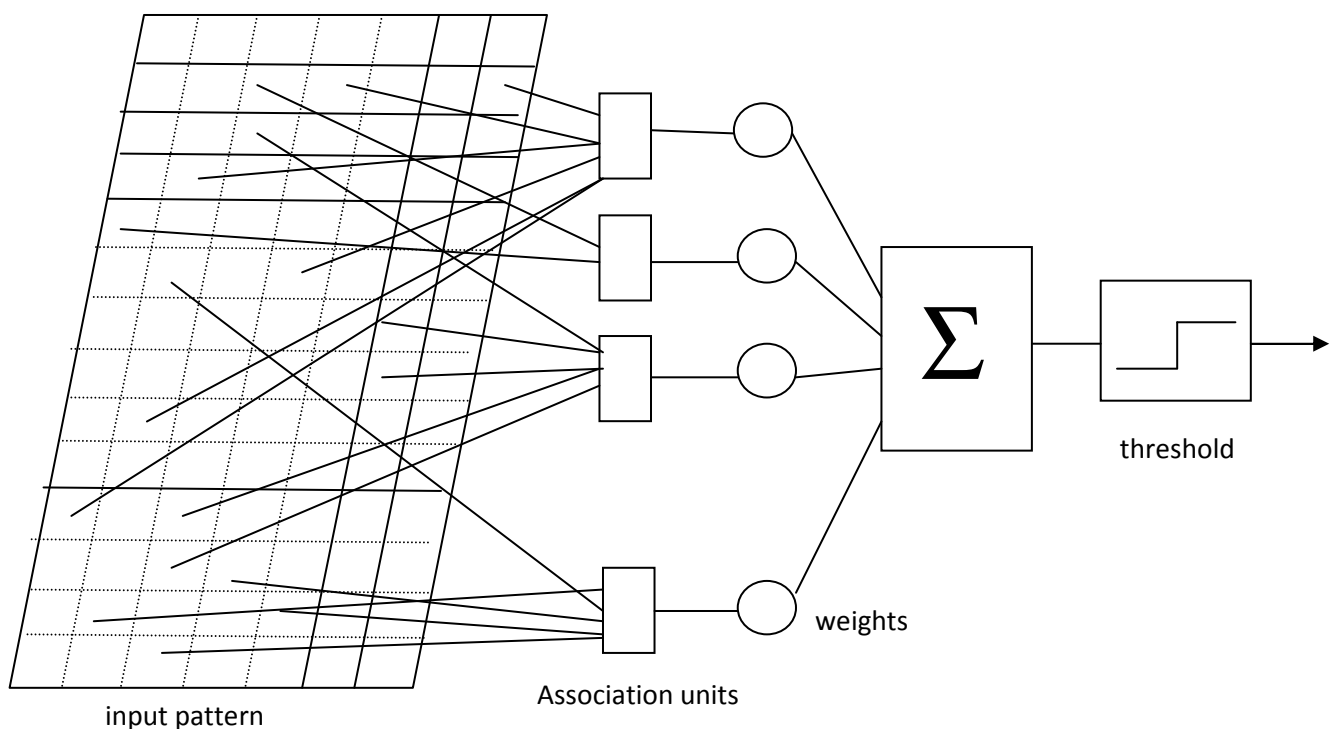
Figure 5.8.A – The Perceptron

5.9 - The Activation Functions

As mentioned previously, the activation function acts as a squashing function, such that the output of a neuron in a neural network is between certain values (usually 0 and 1, or -1 and 1). In general, there are several types of activation functions, denoted by Φ(.), some of them have been seen in Section 5.1, other common types are described follow in this Section.

There is the Threshold Function which takes on a value of 0 if the summed input is less than a certain threshold value (v), and the value 1 if the summed input is greater than or equal to the threshold value.

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

Secondly, there is the Piecewise-Linear function. This function again can take on the values of 0 or 1, but can also take on values between that depending on the amplification factor in a certain region of linear operation.

$$\varphi(v) = \begin{cases} 1 & v \geq \frac{1}{2} \\ v & -\frac{1}{2} > v > \frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases}$$

Thirdly, there is the sigmoid function. This function can range between 0 and 1, but it is also sometimes useful to use the -1 to 1 range. An example of the sigmoid function is the hyperbolic tangent function.

$$\varphi(v) = \tanh\left(\frac{v}{2}\right) = \frac{1 - \exp(-v)}{1 + \exp(-v)}$$
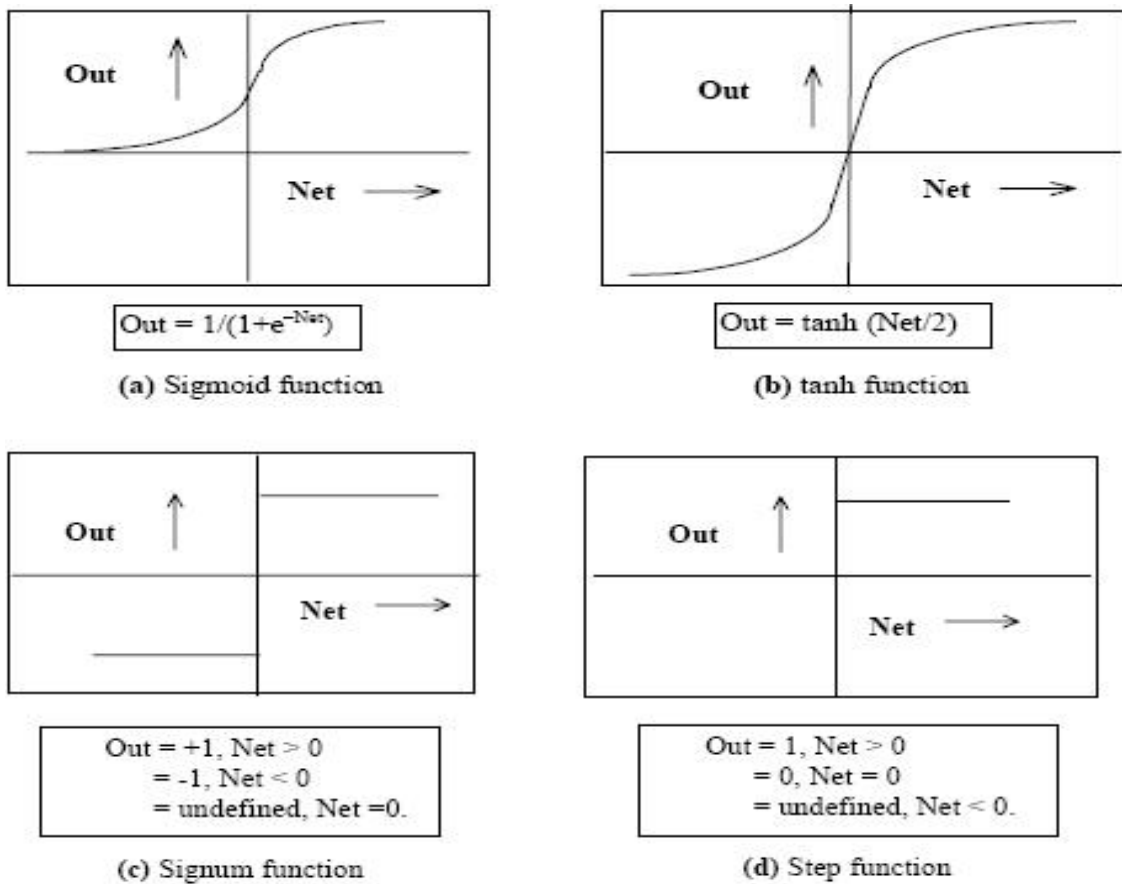
Figure 5.9.A – Common non-linear functions used as activation function. (a) Sigmoid or Logistic and (b) tanh, (c) Signum and (d) step.

5.10 - Single Layer Nets

Using the perceptron training algorithm we are now in position to use a single perceptron or TLU to classify two linearly separable classes A and B. Although the patterns may have many inputs, we may illustrate the pattern space in a schematic or cartoon way as shown in Figure 5.10.A. Thus the two axes are not labeled, since they do not correspond to specific vector components, but are merely indicative that we are thinking of the vectors in their pattern space.
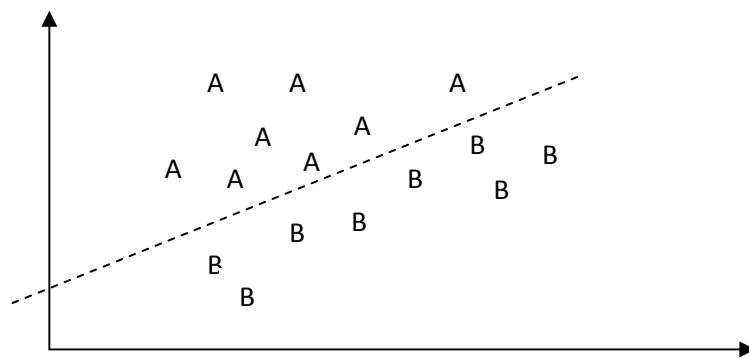
Figure 5.10.A – Classification of two classes A, B.

It is possible, however, to train multiple nodes on the input space to achieve set linearly separable dichotomies of the type shown in the figure 5.10.A. This might occur, for example, if we wish to classify handwritten alphabetic characters where 26 dichotomies are required, each one separating one letter class from the rest of the alphabet, "A"s from non "A"s, "B"s from non "B"s, etc. The entire collection of nodes forms a single-layer net as shown in Figure 5.10.B. Of course, whether each of the above dichotomies is linearly separable is another question. If they are, then the perceptron rule may be applied successfully to each node individually.
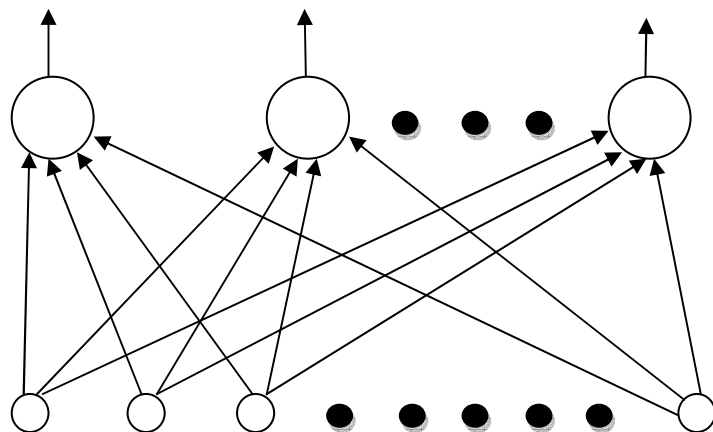
Figure 5.10.B – Single Layer Net

5.11 - How to choose the best Activation Function or TLU

Activation functions for the hidden units[1] are needed to introduce nonlinearity into the network. Without nonlinearity, hidden units would not make nets more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). The reason is that a linear function of linear functions is again a linear function. However, it is the nonlinearity (i.e, the capability to represent nonlinear functions) that makes multilayer networks so powerful. Almost any nonlinear function does the job, except for polynomials. For backpropagation learning, the activation function must be differentiable, and it helps if the function is bounded; the sigmoidal functions such as logistic and tanh and the Gaussian function are the most common choices. Functions such as tanh or arctan that produce both positive and negative values tend to yield faster training than functions that produce only positive values such as logistic, because of better numerical conditioning (see ftp://ftp.sas.com/pub/neural/illcond/illcond.html).

---

[1] **HIDDEN UNITS**: in a neural network that mediate the propagation of activity between input and output layers. The activations or target values of such units are not specified by the environment, but instead arise from the application of a learning procedure that sets the connection weights into and out of the unit.

For hidden units, sigmoid activation functions are usually preferable to threshold activation functions. Networks with threshold units are difficult to train because the error function is stepwise constant, hence the gradient either does not exist or is zero, making it impossible to use backprop or more efficient gradient-based training methods. Even for training methods that do not use gradients--such as simulated annealing and genetic algorithms--sigmoid units are easier to train than threshold units. With sigmoid units, a small change in the weights will usually produce a change in the outputs, which makes it possible to tell whether that change in the weights is good or bad. With threshold units, a small change in the weights will often produce no change in the outputs.

For the output units, we should choose an activation function suited to the distribution of the target values:

- For binary (0/1) targets, the logistic function is an excellent choice   (Jordan, 1995).
- For categorical targets using 1-of-C coding, the softmax activation function is the logical extension of the logistic function.
- For continuous-valued targets with a bounded range, the logistic and tanh functions can be used, provided you either scale the outputs to the range of the targets or scale the targets to the range of the output activation function ("scaling" means multiplying by and adding appropriate constants).
- If the target values are positive but have no known upper bound, we can use an exponential output activation function, but beware of overflow.
- For continuous-valued targets with no known bounds, use the identity or "linear" activation function (which amounts to no activation function) unless we have a very good reason to do otherwise.

There are certain natural associations between output activation functions and various noise distributions which have been studied by statisticians in the context of generalized linear models. The output activation function is the inverse of what statisticians call the "link function".   McCullagh, P. and Nelder, J.A. (1989)

5.12 - Artificial Neural Networks – Real World Applications

The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.
- Robotics, including directing manipulators, computer numerical control.

Application areas include system identification and control (vehicle control, process control), quantum chemistry, game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications (automated trading systems), data mining (or knowledge discovery in databases, "KDD").

5.13 - BPMaster Neural Network Simulator

Neural network simulators are software applications that are used to simulate the behavior of artificial or biological neural networks. They focus on one or a limited number of specific types of neural networks. They are typically stand-alone and not intended to produce general neural networks that can be integrated in other software. Simulators usually have some form of built-in visualization to monitor the training process. Some simulators also visualize the physical structure of the neural network.

BPMaster is a software developed in the Stanford University, and was created by J. L. McClelland and D. E. Rumelhart's Explorations in Parallel Distributed Processing: A handbook of models, programs, and exercises (Cambridge, MA: MIT Press).

After that, in the Universitat Politècnica de Catalunya, was updated with new functionalities like set/ ensayo, "ensayo" or "backsel" by Enrique Romero and J.M. Sopena.

We have used the "ensayo" test, which make three data groups (train, validation and test), using a logistic activation function.

Using the program BPmaster we have developed the neural network training experiment, using the logistic activation function in a simple neural network. The program BPmaster has a command called "ensayo" that allows this type of neural training with options in a number of activation functions.

Once executed the training's the test results are presented as train, validation and test, those are the averaged of the results of each epoch made, from the validation results, we obtain the average neural training results.

At the same time cross-validation is performed, allowing to know the accuracy on the validation test set.

Cross validation is the practice of partitioning the available data into multiple subsamples such that one or more of the subsamples is used to fit (train) a neural model, with the remaining subsamples being used to test how well the model performs. Because the model is tested on data not used in training the model, cross validation can provide a useful estimate of how the model will perform on new data.

The number of cross-validations is the number of times that the whole process of cross-validation is run. Therefore each cross-validation the order of the patterns changes.

You can type in the output file the:

- • Results of each epoch
- • Summary of results of each test
- • Summary of the set of all tests

The results labeled training, validation and test are interpreted as:

1. Training: Results (in training, validation and test sets) using the network giving the optimal training results.
2. Validation: Results (in training, validation and test sets) using the network giving the optimal validation results.
3. Test: Results (in training, validation and test sets) using the network giving the optimal test results.

The first is to see how much overfitting exists. The second is what gives the result of generalization (test results in the optimal validation). The third is an estimate of the best that could have been obtained.
In the end, the means of the results of each test are displayed (this is where we have to look at the result of generalization).

6 - Feature Classification and Selection

Building accurate predictive models in many domains requires consideration of hundreds of thousands or millions of features. Models which use the entire set of features will almost certainly over fit on future datasets. Standard statistical and machine learning methods such as SVMs, maximum entropy methods, decision trees and neural networks generally assume that all features are known in advance. They use regularization (e.g. ridge regression, weight decay in neural networks or Gaussian) or feature selection to avoid over fit.

Feature selection offer many benefits beyond improved prediction accuracy. For example, feature selection can reduce the requirements of measuring and storing data. Non- predictive or redundant expensive features may not need to be collected or captured. Feature selection can also speed up model updating. When large amount of new information are available on a minute by minute basis, the time to update models can be prohibited if large sets of features are involved. Selecting the most important features from the new information will speed up the model updating. Other needs include better data understanding.

This section is focused on selecting a subset of features for spam classification attributes to build a good predictive model for ANN antispam.

6.1 - Feature Selection brief review and definitions

The feature selection model typically assumes a setting consisting of n observations and a fixed number m of candidate features.

The goal is to select the feature subset that will ultimately lead to the best performing predictive model. The size of the search space is therefore 2^m, and identifying the best subset is NP-complete. Many commercial statistical packages offer variants of a greedy method, "forward feature selection", an iterative procedure in which at each step all features are tested and the single best feature is selected and added to the model. Feature Selection performs hill climbing in the space of feature subsets. Feature selection is terminated when either all candidate features have been added, or none of the remaining features lead to increased expected benefit according to some measure, such as a p-value threshold. Variants of stepwise selection abound, including forward (adding features deemed helpful), backward (removing features no longer deemed helpful), and mixed methods (alternating between forward and backward). Our study and work will assume a simple forward search.

There are many methods for assessing the benefit of adding a feature. Computer scientists tend to use cross-validation, where the data set is divided into several (say k) batches with equal sizes. K-1 of the batches are used for training while the remainder batch is used for evaluation. The training procedure is run k times so that the model is evaluated once on each of the batches and performance is averaged. The approach is computationally expensive, requiring k separate training steps for each evaluation. A second disadvantage is that when observations are scarce the method does not make good use of the observations.

A fundamental problem of machine learning is to approximate the functional relationship $f( )$ between an input $X = \{x1, x2, ... ,xM\}$ and an output Y, based on a memory of data points, $\{Xi, Yi\}, i = 1, ..., N$, usually the Xi's are vectors of reals and the Yi's are real numbers. Sometimes the output Y is not determined by the complete set of the input features $\{x1, x2, ... ,xM\}$, instead, it is decided only by a subset of them $\{x(1), x(2), ... , x(m)\}$, where m < M . With sufficient data and time, it is fine to use all the input features, including those irrelevant features, to approximate the underlying function between the input and the output. But in practice, there are two problems which may be evoked by the irrelevant features involved in the learning process.

The irrelevant input features will induce greater computational cost. For example, using cached kd-trees, locally weighted linear regression's computational expense is $O(m^3 + m^2 \log N)$ for doing a single prediction, where N is the number of data points in memory and m is the number of features used. Apparently, with more features, the computational cost for predictions will increase polynomially; especially when there are a large number of such predictions, the computational cost will increase immensely.

The irrelevant input features may lead to overfitting. For example, in the domain of medical diagnosis, our purpose is to infer the relationship between the symptoms and their corresponding diagnosis. If by mistake we include the patient ID number as one input feature, an over-tuned machine learning process may come to the conclusion that the illness is determined by the ID number.

Another motivation for feature selection is that, since our goal is to approximate the underlying function between the input and the output, it is reasonable and important to ignore those input features with little effect on the output, so as to keep the size of the approximator model small.

For example, [Akaike, 73] proposed several versions of model selection criteria, which basically are the trade-offs between high accuracy and small model size.

The feature selection problem has been studied by the statistics and machine learning communities for many years. It has received more attention recently because of enthusiastic research in data mining. According to [John et al., 94]'s definition, [Kira et al, 92] [Almuallim et al., 91] [Moore et al, 94] [Skalak, 94] [Koller et al, 96] can be labelled as "filter" models, while [Caruana et al., 94] [Langley et al, 94]'s research is classified as "wrapped around" methods. In the statistics community, feature selection is also known as "subset selection", which is surveyed thoroughly in [Miller, 90].

The brute-force feature selection method is to exhaustively evaluate all possible combinations of the input features, and then find the best subset. Obviously, the exhaustive search's computational cost is prohibitively high, with considerable danger of overfitting. Hence, people resort to greedy methods, such as forward selection. In this work, we propose three greedier selection algorithms in order to further enhance the efficiency. We use real-world data sets from email world from different users to obtain the accuracy near the 95% of spam detection using Artificial Neural Networks.

<u>Forward Feature Selection Algorithm</u>

In this section, we introduce the conventional feature selection algorithm: forward feature selection algorithm; forward algorithm, in order to improve the computational efficiency without sacrificing too much accuracy.

The forward feature selection procedure begins by evaluating all feature subsets which consist of only one input attribute. In other words, we start by measuring each attribute error for every one-component subsets, *{X1}, {X2}, ..., {XM}*, where *M* is the input dimensionality; so that we can find the best individual feature, *X(1).*

FS:= 0
Candidates:= {1,2,…m} // all features

For j := 1 to m do

For every feature K in candidate do
    Extended FS:= FS U {K}
    Evaluate extended FS using P-partitional dataset and store it in
    BestExtendedFS If it is the best among the subsets teste in this inner loop;
End For

Let K_{Best} be the feature added in BestExtendedFS;
FS:=BestExtendedFS; // FS:= U {K_{Best}}
Candidates := Candidates – {K_{Best}};
Evaluate extended FS using P-partitional dataset and store it in BestFS If it is the best
among the subsets teste in this inner loop;

End For

Return Best FS;

Algorithm 6.3.A – Feed Forward Algorithm for Feature Selection

Procedure evaluate FS using P-partitional Dataset

For each fold (i=0,1,2,.....P-1)
    Let Trainset(i):= all folds except I;
    Let Trainset(i):= the i´th fold;
    Train an ANN_i with TrainSet(i) using FS;
    $TestScore_i$:= RMS score of ANN_i onTestSet(i)
End For

Return the mean Test Score for the P values of TestScore

Algorithm 6.3.B – Cross-validation for P-partitional Dataset

Procedure evaluate FS using Double Cross-Validation and P-partitional Dataset

For each fold (i=0,1,2,.....P-1)
    Let Testset(i):= the i´th fold;
    For each fold j ≠ i
        Let Validationset (i,j) := the j´th fold;
        Let Trainset (I,j) := all folds except i and j;
        Train an ANN_{ij} with Trainset(i,j) and Validationset(i,j) using FS;
        $TestScore_{ij}$:= RMS error of ANN_{ij} on TestSet(i)
    End For
End For

Return the mean TestScore for the P*(P-1) values of $TestScore_{ij}$

Algorithm 6.3.C – Double Cross-validation for P-partitional Dataset

Where the computational cost in single-layer perceptron will be:

$$\text{Computational Cost} = O(P^2 * \text{MaxEpoch}*N*m)$$

The goal of feature selection is to choose a subset X_S of the complete set of input features $X= \{x_1, x_2, \dots, x_M\}$ so that the subset X_S can predict the output Y with accuracy comparable or even better to the performance of the complete input set X, and with great reduction of the computational cost.

First, let clarify how to evaluate the performance of a set of input features. Even if feature sets are evaluated by testset cross-validation or leave-one-out cross validation, an exhaustive search of possible feature-sets is likely to find a misleadingly well-scoring feature-set by chance. To prevent this, we use the cascaded cross-validation procedure in Figure N upper, which selects from increasingly large sets of features (and thus from increasingly large model classes).

To evaluate the performance of a feature selection algorithm is more complicated than to evaluate a feature set. This is because in order to evaluate an algorithm, we must first ask the algorithm to find the best feature subset. Second, to give a fair estimate of how well the feature selection algorithm performs, we should try the first step on different datasets.

6.2 - Feature selection algorithms

Now we will introduce the conventional feature selection algorithm: forward feature selection algorithm; then we explores briefly the variants of the forward algorithm for study porpoise, and briefly we will describe each feature selection variant.

6.2.1 - Exhaustive Search feature selection

To find the overall best input feature set, we can also employ exhaustive search. Exhaustive search begins with searching the best one-component subset of the input features, which is the same in the forward selection algorithm; then it goes to find the best two-component feature subset which may consist of any pairs of the input features. Afterwards, it moves to find the best triple out of all the combinations of any three input features, etc. It is straightforward to see that the cost of exhaustive search is the following:

$$\text{M}C(1) + \binom{\text{M}}{2} C(2) + \cdots + \binom{\text{M}}{\text{m}} C(\text{m})$$

Compared with the exhaustive search, forward selection is much cheaper. Exhaustive search has the most expensive computational cost.

6.2.2 - Backward feature selection "backsel"

Backward elimination or Backward feature selection is a feature search starting with the full feature set. At each iteration the next candidate subsets are formed by eliminating each feature, one at a time, from the current subset. The feature whose elimination resulted in the highest performance improvement is eliminated permanently. The search stops when the termination condition is met or only one feature remains in the set.
Backward feature selection has very high computational cost.

The computational cost is:

BS = 1 C(computational cost(1) + 2 C(computational cost(2)+…. + m C(computational cost(m)

Although the cost is asymptotically similar to the cost of forward feature selection, is not the same computationally, due to not so efficiency compared to Forward Feature Selection, as we will see in the following section.

6.2.3 - Forward feature selection

The forward feature selection procedure begins by evaluating all feature subsets which consist of only one input attribute. In other words, we start by measuring the Cross-Validation (CV) error of the one-component subsets, $\{X_1\}$, $\{X_2\}$, …, $\{X_M\}$, where M is the input dimensionality; so that we can find the best individual feature, $X_{(1)}$.
Next, forward selection finds the best subset consisting of two components, $X_{(1)}$ and one other feature from the remaining M-1 input attributes. Hence, there are a total of M-1 pairs. Let's assume $X_{(2)}$ is the other attribute in the best pair besides $X_{(1)}$.

Afterwards, the input subsets with three, four, and more features are evaluated. According to forward selection, the best subset with m features is the m-tuple consisting of $X_{(1)},X_{(2)}$, …, $X_{(m)}$, while overall the best feature set is the winner out of all the M steps. Assuming the cost of a CV evaluation with i features is C(i), then the computational cost of forward selection searching for a feature subset of size m out of M total input attributes will be

$$MC\ (1)+ (M-1)C\ (2)+...+ (M-m+1)C(m)$$

For example, the cost of one prediction with ANN using Single-Layer Perceptron will be:

Computational Cost = O (M^2 * C(evaluation cost FS))

Or

Computational Cost = O (M^3 * P^2 * Max_{epoch} * N)

Where M^3 is the total of candidates and P^2 is the P partitioned dataset and M(epoch) the

Maximum quantity of epoch used by the ANN and N the number of datapoints.

In others words

FS = m C(computational cost(1) + m-1 C(computational cost(2)+…. + 1 C(computational cost(m)

However, forward selection may suffer because of its greediness. For example, if $X_{(1)}$ is the best individual feature, it does not guarantee that either $\{X_{(1)}, X_{(2)}\}$ or $\{X_{(1)}, X_{(3)}\}$ must be better than $\{X_{(2)}, X_{(3)}\}$. Therefore, a forward selection algorithm may select a feature set different from that selected by exhaustive searching. With a bad selection of the input features, the prediction Y_q of a query $X_q = \{x_1, x_2, \dots, x_M\}$ may be significantly different from the true Y_q.

6.2.4 - Restricted Forward Selection (RFS)

1- Calculate all the 1-feature set using any classification techniques to obtain the errors, and sort the features according to the corresponding errors. Suppose the features ranking from the most important to the least important are $X_{(1)}, X_{(2)}, \dots, X_{(M)}$.

2- Do the classification techniques of 2-feature subsets which consist of the winner of the first round, $X_{(1)}$, along with another feature, either $X_{(2)}$, or $X_{(3)}$, or any other one until $X_{(M/2)}$. There are $M/2$ of these pairs. The winner of this round will be the best 2-component feature subset chosen by RFS.

3- Calculate using the classification techniques to obtain errors of $M/3$ subsets which consist of the winner of the second round, along with the other $M/3$ features at the top of the remaining rank. In this way, RFS will select its best feature triple.

4- Continue this procedure, until RFS has found the best m-component feature set.

5- From Step 1 to Step 4, RFS has found m feature sets whose sizes range from 1 to m. By comparing their errors, RFS can find the best overall feature set.

The difference between RFS and conventional Forward Selection (FS) is that at each step to insert an additional feature into the subset, FS considers all the remaining features, while RFS only tries a part of them which seem more promising.

6.2.5 - Greedy Algorithm

Do all the 1-attribute using any classification techniques and sort them, take the best two individual features and evaluate their error, then take the best three individual features, and so on, until m features have been evaluated. Compared with the super greedy algorithm, this algorithm may conclude at a subset whose size is smaller than m but whose inner testset error is smaller than that of the m-component feature set. Hence, the greedy algorithm may end up with a better feature set than the super-greedy one does.

6.2.6 - Super Greedy Algorithm

Do all the 1-attribute using any classification techniques calculations, sort the individual features according to their mean error, then take the m best features as the selected subset. We thus do M computations involving one feature and one computation involving m features.

After studying all forms of feature selection described in the previous sections, now we conducted the experiments using the feed forward feature selection algorithm with Single-Layer Perceptron Artificial Neural Network.

This algorithm delivers a level of acceptable compromise between the computational cost and quality of search features. As we have seen the rest of algorithms such as Backward has a very high computational cost, or as the Greedy algorithms that although it's computational cost is very low, the accuracy of the features it provides is highly questionable.

The forward selection used in this work has been the Forward Feature Selection. We resort to experiments using real world spam dataset for spam and ham classification. You may see their results in the following chapter.

7 - Dataset Creation "the email corpus"

In this project we have used a corpus of 2200 e-mails received from different senders to different receivers collected by the ISP (www.masbytes.com in Spain) over a period of several months. Some of the e-mails contained embedded attachments.

The corpus was hand (visual-human review) checked to classify spam and ham, follow-on 1100 ham and 1100 spam.

Each e-mail message was saved as a text file or HTML, and then parsed to identify each header element (such as Received: or Subject:) to distinguish them from the body of the message.

We have taken several attributes or features from those emails, every attribute has been obtained based in our expertise looking how spammers are making the email undetectable, then we have defined some features for in spam recognition(using for example, antivirus, char, vowels, etc).
The objective was to improve the feature selection using not only email attribute but also characterization using opensource software as SPF, ClamAV, Whitelist, blacklist, etc. You may see more references in the first section of this work.

7.1 – How we have obtained "the corpus"

Due to the large amount of anti-spam filters available today, results of performance testing have been studied separately within the open source community to verify the validity of each one of them.

Therefore the filters implemented and executed, have a high probability to be very valid and secure when detect both spam and ham.

Here you will find a detailed description of the filters used. For more information, please visit sourceforge.net.

Charts blots: Unlike other antispam systems, lists may apply SINESPAM blots of IP before any other filter. This ensures that the client can receive mail from specific servers but these servers are labeled as disreputable.

Ips Reputation: The second layer of filtering is the reputation of IP and RBLs. It checks the reputation of the source server so that by studying their behavior, both historical and current, you can categorize your mail and eventually eliminate up to 90% of spam. Not only was immensely successful in reducing the amount of spam but it makes the most efficient way to close the connection to the spammer even before receiving the mail. In these cases, the spammer detects that you are not accepted mail and takes it into account when deciding on the least protected domains to which address the following attacks. To get the number of false positives is virtually zero SINESPAM leaving no mail is not at least in two of the six RBLs consulted.

Charts blots by domain or email address: Both the manager and the user can enter addresses and domains known to be sure they will not be filtered by creating false positives.

Lists of trust: trust lists are automatically with valid mailing addresses that regularly receives each user. This list is personal and is generated automatically, using a proprietary algorithm to SINESPAM that ensures the reliability of the accounts. Thanks to the trust lists avoiding false positives without the user having to intervene at any time.

SPF: Sender Policy Framework Using is achieved to ensure that the servers from which SINESPAM receives the mail are allowed to send mail from a specific domain. This technique prevents email spoofing, or phishing. In order to use SPF, it must be properly configured on the source servers.

Razor Servers: spam-catcher using a collaborative filtering network to Vipul's Razor is in distributed, collaborative, spam detection and filtering network. Establishes a Razor in distributed and Constantly updating catalog of spam in propagation. This catalog is Used by clients to filter out spam Known. On Receiving a spam, a Razor Reporting Agent (run by an end-user or a troll box) and submits Calculates a 20-character unique identification of the spam (a SHA Digest) to ITS Closest Razor Catalogue Server.

SenderDomval: It verifies the existence of the domain of the sender to remove the spam that is sent from nonexistent domains.

Sendercallout: It verifies the existence of the sender to delete the spam that is sent from nonexistent accounts.

Greylisting: The emails are categorized according to the probability that they are valid. When the score they receive does not ensure that the emails are valid, they can apply greylisting technique of allowing a temporary error to the sending server. If the server is sending spam, not normally retried, while if the mail is valid, the server must (if properly configured) to retry sending after some time.

Delay: The emails are categorized according to the probability that they are valid. When the score they receive does not ensure that the emails are valid, you can apply a certain delay in the connection that criminalizes the sending server. If it is a spam server does not want to waste time and cut the connection to try other servers.

Content Filter: A way to customize the filtering is to allow the administrator and individual users the ability to create their own filters. These filters can be created by a company or user and work analyzing the words mail and acting according to its settings.

Bogofilter and Spamassassin: Two of the modules that are used in SINESPAM Bogofilter and Spamassassin. These are based on different antispam techniques ranging from Bayesian filters to DNS-based tests or consultations in Databases. The rules and evidence contained in these two systems are constantly adjusted for optimum performance depending on the type of spam. Some of the tests performed are:

> **Inspection of "Headers":** The "Headers" or message headers contain important information about it.
> **Analysis of the Message**: The body of the message and title are read by SpamAssassin, by searching by keywords or structures that make up a spam email.
> **Analysis probabilistic / Bayesian**: Once you've set the initial rules for the detection, probabilistic analysis was performed to determine similarities between incoming messages and those already previously identified as SPAM.

Lists "Hash" Signatures Mail: Because a SPAM email usually sent thousands of people at once, the structure of each message is identical in all instances, thus producing a "Hash" unequivocal. SpamAssasisn query lists of "hashes" on messages known.

Antivirus: Virus scanning is applied to all emails within the system, whether valid or is considered spam. The virus is constantly updated automatically. There is the possibility to disable anti-virus filtering from the control panel. (SpamCop and CLAM-AV)

The definition of the 31 attributes initially proposed is given in the following table.

7.2 - Features from the Message Body and Header

Attributes definition for the emails collected, to use in the learning system based on Artificial Neural Nets.

We have two types of components within the email, the first is the header (which defines the SMTP protocol headers) and the second body of the email (where email drafted Itself).

The following table shows that to detect patterns in email spam, you must search or inspect both the email header as in the body.

The Table is divided into two groups header and body, in the first column is the description of the attribute to be inspected in the next column is the value of the variable definition for the email corpus and the third column definition the variable itself.

NOTE: Some of the attributes described in the following table not have been used in extracting the information contained the values in themselves, because they had no certain and quantifiable results.

Feature Definition	Value	Variable
HEADER		
Attribute Spamassasin – Verify email has passed spamassasin spam= 1 nospam=0	Binary YES=1 NO=0	att_spamassasin
Sender Blacklisted rbl=1 norbl=0	Binario YES=1 NO=0	att_rbl
Attribute char extended – char extended quantity: ¡ , ! , · , # , $,€ , %, ~, &, / , (,), = , ¿, ?, *, ^, { , }, [,] ,º, \ , @, >, < ¿ , ´,´ , +, - , _ , ¨,	Number = Char Quantity	att_char_extend
Verify the name sender has only alphabetic words yes=1 or no= 0	Binary YES=1 NO=0	Not corresponding bad feature
Verify the name sender has only vowels yes = 1 or no= 0	Binary YES=1 NO=0	Not corresponding bad feature
Number of alphabetic words that did not contain any vowels on Header	Number = Char Quantity	att_char_header_consonante
Number of alphabetic words that contained at least two of the following letters (upper or lower case): K, Q, X, Z, Y, W,V on header	Number = Char Quantity	att_char_nospanish

Number of alphabetic words that were at least 15 characters long – Not URL	Number = Char Quantity	att_long_words
Number of words with all alphabetic characters in upper case	Number = Char Quantity	att_char_alluppercase
Verify SPF Sender tiene spf=1 nospf=0	Binary YES=1 NO=0	att_spf
Binary feature indicating whether a priority header appeared within the message headers (X-Priority and/or X-MSMail-priority) or whether the priority had been set to any level besides normal or medium: yes = 1, no = 0	Binary YES=1 NO=0	att_header_priority
Binary feature indicating whether a content-type header appeared within the message headers or whether the content type of the message has been set to "text/html": yes = 1, no = 0 Features From the Message Body	Binary YES=1 NO=0	att_header_html
Verify the Sender from valid MX -	Binario YES=1 NO=0	att_mx_sender
Verify Virus email – passed by the clamAV antivirus	Binary YES=1 NO=0	att_virus
Verify email has passed OK by the sanesecurity spam database spam=1 nospam=0	Binary YES=1 NO=0	att_sane_spam
Verify email has passed OK by the sanesecurity virus database virus=1 novirus=0	Binary YES=1 NO=0	att_sane_virus
Verify RAZOR Sender yes=1 no=0	Binary YES=1 NO=0	att_razor
Verify PYZOR Sender yes=1 no=0	Binary YES=1 NO=0	att_pyzor
Verify DCC Sender dcc=1 nodcc=0	Binary YES=1 NO=0	att_dcc
BODY		
Number of URLs within hyperlinks that contain any numeric digits or any of three special characters ("&", "%" or "@") in the domain or subdomain(s) of the link	Number = URLs Hiperlink Quantity	att_body_href_extend_char
Verify the message header subject has been signed by a real sender. – DKIM	Binary YES=1 NO=0	att_dkim
Binary feature indicating occurrence of a character (including spaces) that is repeated at least three times in succession: yes = 1, no = 0 Features From the Priority and Content-Type Headers	Binary YES=1 NO=0	att_char_repeat
Char word extended inside word like V1agr3 – v1agra – character number or extended character inside word without spaces	Number = Char Quantity	att_char3_number
Attribute Spanish – How many spanish letters has the email: á , é , í , ó , ú , Á , É , Í , Ó, Ú, ñ , Ñ	Number = Char Quantity	att_spanish
Verify the email has passed clean by the CRM 114 antispam system with dictionary	Binary YES=1 NO=0	att_crm114
The run-length attributes (55-57) measure the length of sequences of consecutive capital letters - OJO Not URL Not emails	Number = Char Quantity	Not corresponding bad feature

Continuous integer [1,...] attribute of type letters_run_length_longest = length of longest uninterrupted sequence of letters without any space between.	Number = Char Quantity	Not corresponding bad feature
Number of clickable images represented in the HTML	Number = images quantity	att_body_href_img
Binary feature indicating whether a color of any text within the body message was set to white: 1 = yes, 0 = no	Binario YES=1 NO=0	att_body_color
Sender Blacklisted rbl=1 norbl=0	Binario YES=1 NO=0	att_rbl
Number of HTML opening comment tags 14 Number of hyperlinks ("href=")	Number = href Quantity	att_body_href
Number of clickable images represented in the HTML	Number = images quantity	att_body_href_img
URL Link in the Body url=1 nourl=0	Binario YES=1 NO=0	att_count_url

8 - Experimental Results

In this section we apply the Forward Feature selection techniques using the email corpus, and single-layer Artificial Neural Networks as classifiers.

The first step was to identify the best performance classification using ANN with Linear and Logistic Activation Function. The BPMaster program "ensayo" has the possibility to identify three types of Accuracy:

1. Training: Results (in training, validation and test sets) using the network giving the optimal training results.
2. Validation: Results (in training, validation and test sets) using the network giving the optimal validation results.
3. Test: Results (in training, validation and test sets) using the network giving the optimal test results.

For each feature we have running the classification test, using BPMaster, to evaluate the best performance for all features using Cross Validation with 5 and 10 Fold.

The algorithm run for this experimental Classification is described on the Chapter 6. The Algorithm used is on the reference Algorithm 6.3.A togheter with the Algorithm 6.3.C to obtain the following results.

The method used to make the classification experiments, was the Forward Feature Selection, using a Single-Layer ANN as classificator with double Cross-Validation using 5 Fold.

The parameters for ANN described for the experiments has been as follow:

Epoch: 400

Momentum: 0.001

Activation Function: lgt (logistic)

Weight: 0.01

Learning rate: 0.001

Bias: 0.03

Cross-Validation: 5 folds

Below you can see the results and selected features for 5 Cross-validation fold and the results for each experiment from 1 to 27 features.

Features Training by ANN with 5 CV-Fold

Feature	Cross-Validation 5 fold		
	Accuracy Train	Accuracy Validation	Accuracy Test
att_spanish	53,81%	53,90%	53,74%
att_long_words	67,14%	67,14%	67,14%
att_header_priority	67,32%	67,32%	67,32%
att_char_extend	82,27%	82,29%	82,25%
att_char_nospanish	80,55%	80,55%	80,55%
att_char_alluppercase	58,53%	58,53%	58,48%
att_spamassasin	52,41%	52,41%	52,41%
att_body_href	69,77%	69,77%	69,77%
att_crm114	62,91%	62,91%	62,91%
att_virus	55,59%	55,59%	55,59%
att_sane_spam	68,91%	68,91%	68,91%
att_sane_virus	54,36%	54,36%	54,36%
att_razor	57,73%	57,73%	57,73%
att_dcc	68,64%	68,64%	68,64%
att_vocal	50,64%	50,64%	50,64%
att_char3_number	81,65%	81,66%	81,65%
att_char_header_consonante	52,02%	52,02%	52,04%
att_header_html	74,68%	74,68%	74,68%
att_mx_sender	51,86%	51,86%	51,86%
att_body_href_img	65,73%	65,73%	65,73%
att_body_color	72,77%	72,77%	72,77%
att_body_href_extend_char	75,23%	75,23%	75,23%
att_dkim	51,91%	51,91%	51,91%
att_count_url	51,86%	51,86%	51,86%
att_spf	51,85%	51,86%	51,86%
att_pyzor	49,98%	50,27%	49,90%
att_char_from_consonante	50,04%	49,85%	50,02%

Artificial Neural Networks definition parameters

Attribute	epochs	momentum	activation function	cv-fold	weight rate	learning rate	bias	Accuracy
att_spanish	400	0.001	lgt	5	0.01	0.001	0.03	53,90%
att_long_words	400	0.001	lgt	5	0.01	0.001	0.03	67,14%
att_header_priority	400	0.001	lgt	5	0.01	0.001	0.03	67,32%
att_char_extend	400	0.001	lgt	5	0.01	0.001	0.03	82,29%
att_char_nospanish	400	0.001	lgt	5	0.01	0.001	0.03	80,55%

att_char_alluppercase	400	0.001	lgt	5	0.01	0.001	0.03	58,53%
att_spamassasin	400	0.001	lgt	5	0.01	0.001	0.03	52,41%
att_body_href	400	0.001	lgt	5	0.01	0.001	0.03	69,77%
att_crm114	400	0.001	lgt	5	0.01	0.001	0.03	62,91%
att_virus	400	0.001	lgt	5	0.01	0.001	0.03	55,59%
att_sane_spam	400	0.001	lgt	5	0.01	0.001	0.03	68,91%
att_sane_virus	400	0.001	lgt	5	0.01	0.001	0.03	54,36%
att_razor	400	0.001	lgt	5	0.01	0.001	0.03	57,73%
att_dcc	400	0.001	lgt	5	0.01	0.001	0.03	68,64%
att_vocal	400	0.001	lgt	5	0.01	0.001	0.03	50,64%
att_char3_number	400	0.001	lgt	5	0.01	0.001	0.03	81,66%
att_char_header_consonante	400	0.001	lgt	5	0.01	0.001	0.03	52,02%
att_header_html	400	0.001	lgt	5	0.01	0.001	0.03	74,68%
att_mx_sender	400	0.001	lgt	5	0.01	0.001	0.03	51,86%
att_body_href_img	400	0.001	lgt	5	0.01	0.001	0.03	65,73%
att_body_color	400	0.001	lgt	5	0.01	0.001	0.03	72,77%
att_body_href_extend_char	400	0.001	lgt	5	0.01	0.001	0.03	75,23%
att_dkim	400	0.001	lgt	5	0.01	0.001	0.03	51,91%
att_count_url	400	0.001	lgt	5	0.01	0.001	0.03	51,86%
att_spf	400	0.001	lgt	5	0.01	0.001	0.03	51,86%
att_pyzor	400	0.001	lgt	5	0.01	0.001	0.03	50,27%
att_char_from_consonante	400	0.001	lgt	5	0.01	0.001	0.03	49,85%

In the first part of the algorithm gets the value of each FS (Feature), and the accuracy itself, showing that not all FS (Features) have the same accuracy.

As the proposed objective was to obtain a 95% or more accurately, using all the features of the data set using the ANN based classifier, and using linear or logistic algorithms. The next step is to experiment with every feature to end all the features and performance rating.

Results using 1 feature – 82,25%

Attribute	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	Ensayo1	82,27%	82,29%	82,25%

Results using 2 features – 82,46%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%	Ensayo2	82,46%	82,47%	82,46%
att_char3_number	81,87%				

Results using 3 features – 82,49%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%	Ensayo3	82,49%	82,50%	82,49%
att_char3_number	81,87%				
att_char_nospanish	80,05%				

Results using 4 features – 82,35%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%	Ensayo4	82,46%	82,35%	82,35%
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				

Results using 5 features – 82,76%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%	Ensayo5	82,84%	82,79%	82,76%
att_body_href_extend_char	75,23%				
att_body_color	72,77%				

Results using 6 features – 82,79%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%	Ensayo6	82,99%	82,79%	82,75%
att_body_color	72,77%				
att_body_href	69,75%				

Results using 7 features – 84,32%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%	Ensayo7	84,64%	84,32%	84,32%
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				

Results using 8 features – 84,26%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%	Ensayo8			
att_char_nospanish	80,05%				

Attribute	Accuracy				
att_body_href_extend_char	75,23%				
att_body_color	72,77%		84,62%	84,27%	84,26%
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				

Results using 9 features – 85,39%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%	Ensayo9	86,11%	85,40%	85,39%
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				

Results using 10 features – 86,86%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%	Ensayo10	87,41%	86,86%	86,86%
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				

Results using 11 features – 88,35%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%	Ensayo11	88,51%	88,35%	88,35%
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				

Results using 12 features – 89,38%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%	Ensayo12	89,71%	89,40%	89,38%
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				

Results using 13 features – 91,45%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%	Ensayo13	92,12%	91,46%	91,45%
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%				

Results using 14 features – 94,83%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%	Ensayo14	95,08%	94,84%	94,83%
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%				
att_char_alluppercase	58,53%				

Results using 15 features – 94,75%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%	Ensayo15	95,01%	94,76%	94,75%
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%				
att_char_alluppercase	58,53%				
att_razor	57,73%				

Results using 16 features – 95,96%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%	Ensayo16			
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%		96,34%	95,96%	95,96%
att_body_href_img	65,73%				

Attribute	Accuracy			
att_crm114	62,91%			
att_header_html	62,91%			
att_char_alluppercase	58,53%			
att_razor	57,73%			
att_virus	55,59%			

Results using 17 features – 95,93%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%	Ensayo17	96,30%	95,94%	95,93%
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%				
att_char_alluppercase	58,53%				
att_razor	57,73%				
att_virus	55,59%				
att_sane_virus	54,36%				

Results using 18 features – 96,06%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%	Ensayo18			
att_char_nospanish	80,05%				

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%		96,44%	96,06%	96,06%
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%				
att_char_alluppercase	58,53%				
att_razor	57,73%				
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				

Results using 19 features – 96,22%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%	Ensayo19			
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%		96,70%	96,23%	96,22%
att_header_html	62,91%				

Attribute	Accuracy
att_char_alluppercase	58,53%
att_razor	57,73%
att_virus	55,59%
att_sane_virus	54,36%
att_spanish	53,90%
att_spamassasin	52,41%

Results using 20 features – 96,21%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%	Ensayo20			
att_crm114	62,91%				
att_header_html	62,91%		96,72%	96,21%	96,21%
att_char_alluppercase	58,53%				
att_razor	57,73%				
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				
att_spamassasin	52,41%				
att_char_header_consonante	52,02%				

Results using 21 features – 96,18%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%	Ensayo21			
att_crm114	62,91%		96,67%	96,21%	96,18%
att_header_html	62,91%				
att_char_alluppercase	58,53%				
att_razor	57,73%				
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				
att_spamassasin	52,41%				
att_char_header_consonante	52,02%				
att_dkim	51,91%				

Results using 22 features – 96,19%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%	Ensayo22			
att_body_href_extend_char	75,23%				

Attribute	Accuracy				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%				
att_char_alluppercase	58,53%				
att_razor	57,73%		96,74%	96,19%	96,19%
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				
att_spamassasin	52,41%				
att_char_header_consonante	52,02%				
att_dkim	51,91%				
att_count_url	51,86%				

Results using 23 features – 96,21%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%	Ensayo23			
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%		96,76%	96,21%	96,21%
att_char_alluppercase	58,53%				
att_razor	57,73%				
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				
att_spamassasin	52,41%				
att_char_header_consonante	52,02%				
att_dkim	51,91%				
att_count_url	51,86%				
att_mx_sender	51,86%				

Results using 24 features – 96,18%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%	Ensayo24			
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%		96,70%	96,20%	96,18%
att_char_alluppercase	58,53%				
att_razor	57,73%				

Attribute	Accuracy
att_virus	55,59%
att_sane_virus	54,36%
att_spanish	53,90%
att_spamassasin	52,41%
att_char_header_consonante	52,02%
att_dkim	51,91%
att_count_url	51,86%
att_mx_sender	51,86%
att_spf	51,86%

Results using 25 features – 96,19%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%	Ensayo25			
att_crm114	62,91%				
att_header_html	62,91%				
			96,76%	96,19%	96,19%
att_char_alluppercase	58,53%				
att_razor	57,73%				
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				
att_spamassasin	52,41%				
att_char_header_consonante	52,02%				

Attribute	Accuracy
att_dkim	51,91%
att_count_url	51,86%
att_mx_sender	51,86%
att_spf	51,86%
att_char_extend	82,29%

Results using 26 features – 96,15%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%		96,73%	96,17%	96,15%
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%	Ensayo26			
att_char_alluppercase	58,53%				
att_razor	57,73%				
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				
att_spamassasin	52,41%				
att_char_header_consonante	52,02%				
att_dkim	51,91%				
att_count_url	51,86%				
att_mx_sender	51,86%				
att_spf	51,86%				

Attribute	Accuracy
att_vocal	50,64%
att_char_from_consonante	49,85%

Results using 27 features – 96,10%

Attribute	Accuracy	Ensayo	Accuracy Train	Accuracy Validation	Accuracy Test
att_char_extend	82,29%				
att_char3_number	81,87%				
att_char_nospanish	80,05%				
att_body_href_extend_char	75,23%				
att_body_color	72,77%				
att_body_href	69,75%				
att_sane_spam	68,92%				
att_dcc	68,64%				
att_long_words	68,57%				
att_header_priority	67,32%				
att_body_href_img	65,73%				
att_crm114	62,91%				
att_header_html	62,91%				
att_char_alluppercase	58,53%	Ensayo27			
att_razor	57,73%		96,73%	96,10%	96,10%
att_virus	55,59%				
att_sane_virus	54,36%				
att_spanish	53,90%				
att_spamassasin	52,41%				
att_char_header_consonante	52,02%				
att_dkim	51,91%				
att_count_url	51,86%				
att_mx_sender	51,86%				
att_spf	51,86%				
att_vocal	50,64%				
att_pyzor	49,90%				
att_char_from_consonante	49,85%				

8.1 – Results summarized for every added feature

As a result of experiments, we have obtained the goal of classification of spam, with 95% accuracy, on this basis of testing, may conduct more experiments in the future for further improvement in the features selection. Whether using SVM (Support Vector Machines) or MLP (Multi-Layer Perceptron).

As can be seen from the graph the learning curve reaches their maximum capacity ranking with only 17 of the 28 features used.

Recall that in the first definition of features, 31 features were studied but only 28 were programmed to perform the feature selection experiments.

With this we can show that for a proper features search, able to make a generalization is necessary work on features selection to allow as much accuracy as possible with the least computational cost.

Make a selection feature generalization Improve the performance by a reduction of the dimensionality and by eliminating irrelevant variables.
Given a set of features, select a subset that performs best under the evaluation criteria, in our case using single-layer artificial neural networks as Classifiers.

Our investigation shows that the forward feature selection algorithms improves the efficiency with not too much computational cost, while does not corrupt the correctness of the selected feature data set so that the prediction accuracy using the algorithm have made the experimental results remains satisfactory.

Features Classification Results
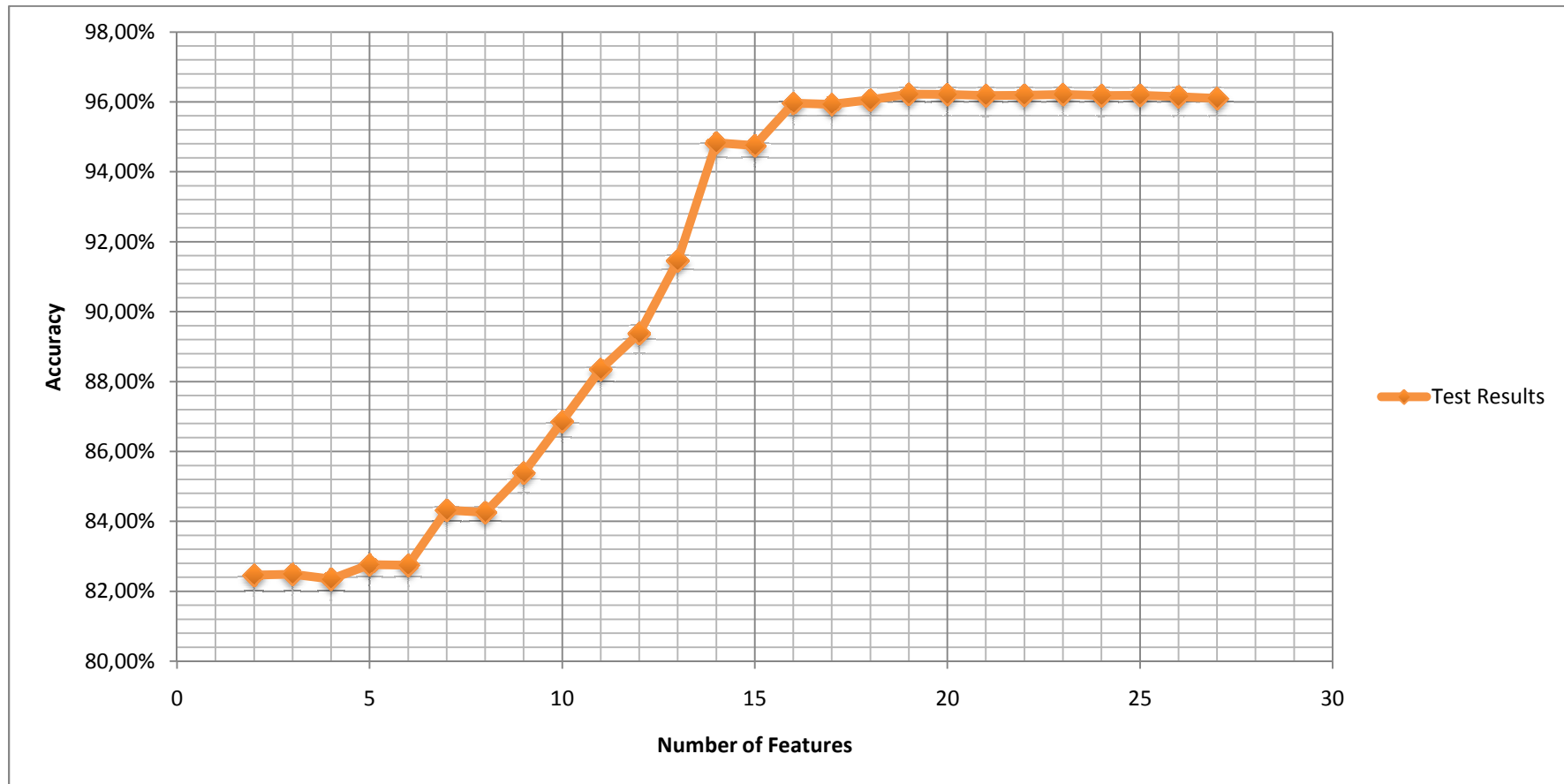
9 – Discussions, Conclusions & Future work

For "the corpus" has been necessary to install a complex system within an ISP in Spain, to collect a large number of "real" emails that recreate in the laboratory of what is happening in the e-mails in Spain and to detect and visualize both the emails themselves, and the structure of spams and hams, to learn and study the characteristics of the email, head and body.

The corpus used in the experiments is 100% real different from the corpus used by other papers studied and annexed in the references of this work, these corpus, have a single sender or recipient. The universe used in the laboratory experiments is 100% real and represents "real" emails where senders and recipients are quite different.

An anti-spam filtering system was proposed which uses the artificial neural network trained by the backpropagation algorithm. The results clearly show that the Subject and Body fields can contain enough information for spam classification in order to obtain near 95% prediction accuracy.

Although the NN technique is accurate and useful, its spam precision performance is not high enough for it to be used without supervision. For this technique to be more useful, the feature set would require additional members or modifications. It should be noted, however, that the NN required fewer features to achieve results similar to the Naïve Bayesian approaches, indicating that descriptive qualities of words and messages, similar to those used by human readers, can be used effectively to distinguish spam by a classifier.

For future work, we suggest that a combination of keywords and descriptive characteristics may provide more accurate classification, as well as the combine of spam classificators techniques. Also we pretend to implement this work in the real world using this features and ANN trained to see how is the potential for spam classification and how to improve the accuracy near 99% and the CPU performance.

A neural network classifier using these descriptive features, however, may not degrade over time as rapidly as classifiers that rely upon a relatively static vocabulary from spammers. Strategies that apply a combination of techniques, such as a NN with whitelist, would likely yield better results.

We have interest in continuing professional development of this thesis work, incorporating new features that provide artificial neural networks, as well as a self-learning system that allows work in the real world on the basis of new features and classification techniques spam The ultimate goal would be to obtain a 99,9% of maximum accuracy.

Also incorporate other classification methods such as SVM (Support Vector Machines) or MLP (Multi-Layer Perceptron) enabling new studies that would achieve more accurate spam classification, as we said our goal is to obtain 99,9% accuracy.

10 – Bibliography References

1- Abduelbaset M. Goweder, Tarik Rashed , Ali S. Elbekaie, and Husien A. Alhammi, "An anti-spam system using artificial neural networks and genetic algorithms", Proc. Int. Arab Conf. on Information Technology, 2008.

2- Akaike:1973 - H. Akaike. Information theory as an extension of the maximum likelihood principle. In B.N. Petrov and F. Csaki, editors, *Second International Symposium on Information Theory*, 267-281, Akademiai Kiado, Budapest, 1973.

3- Alexandru Catalin Cosoi, "A False Positive Safe Neural Network; The Followers of the Anatrim Waves", Proc. Spam Conference 2008.

4- Almuallim et al, 91 - H.Almuallim, and T.G.Dietterich. Learning with many irrelevant features. In Porc. AAAI-91, pp 547-552. MIT Press, 1991.

5- Androutsopoulos, I; Koutsias, J; Chandrinos, K. V. and Spyropoulos, C. D. 2000. An experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (Athens, Greece, 2000), pp. 160-167.

6- Bishop, Christopher M. (1996-01-18). *Neural Networks for Pattern Recognition*. Oxford University Press. p. 504. ISBN 0198538642 ISBN 978-0198538646. http://research.microsoft.com/~cmbishop/books.htm.

7- Bishop, Christopher M. (2006-08-17). *Pattern Recognition and Machine Learning*. Springer. p. 738. ISBN 978-0387310732. http://research.microsoft.com/~cmbishop/PRML.htm.

8- Burton, B. 2002. SpamProbe – Bayesian Spam Filtering Tweaks. http://spamprobe.sourceforge.net/paper.html; last accessed November 17, 2003.

9- C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller (1996). "**Context-specific independence in Bayesian networks**." *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI '96)* (pp. 115-123).

10- Caruana et al, 94] R. Caruana, and D. Freitag. Greedy attribute selection. In Proc. ML-94, Morgan Kaufmann, 1994.

11- Cranor, L. F. and LaMacchia, B. A. 1998. Spam! Communications of the ACM, 41(8): pp. 74-83.

12- Declude, IP Lookup Against a List of All Known DNS-based Spam Databases. http://www.declude.com/junkmail/support/ip4r.htm; last accessed January 27, 2004.

13- Graham, P. 2002. A Plan for Spam. http://www.paulgraham.com/spam.html; last accessed November 17, 2003.

14- Graham, P. 2003. Better Bayesian Filtering. In Proceedings of the 2003 Spam Conference (Cambridge, Massachusetts,2003). See http://spamconference.org/proceedings2003.html.

15- Hauser, S. 2002. Statistical Spam Filter Works for Me. http://www.sofbot.com/article/Statistical_spam_filter.html; last accessed November 17, 2003.

16- Hauser. S. 2003. Statistical Spam Filter Review. http://www.sofbot.com/article/Spam_review.html; last accessed November 17, 2003.

17- Haykin, 1994 - Haykin, S. (1994). Neural networks: a comprehensive foundation. Macmillan, New York.

18- James Clark, Irena Koprinska, and Josiah Poon, "A Neural Network Based Approach to Automated E-mail Classification", Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence, IEEE Computer Society.

19- John(1994) G. H. John.
When the best move isn't optimal: Q-learning with exploration.
In Proceedings of the Twelfth National Conference on Artificial Intelligence, page 1464, Seattle, WA, 1994.

20- Jordan, M.I. (1995), "Why the logistic function? A tutorial discussion on probabilities and neural networks", MIT Computational Cognitive Science Report 9503, http://www.cs.berkeley.edu/~jordan/papers/uai.ps.Z.

21- Kira et al, 92 K.Kira, and L.A.Rendell, The feature selection problem: traditional methodsand a new algorithms. In Proc. AAAI-92, pp 129-134. MIT Press. 1992

22- Langley et al, 94 - P.Langley, and S.Sage, Induction of selective Bayesian classifiers. In Proc.UAI-94, pp 399-406. Seattle, WA. Morgan Kaufmann, 1994.

23- McCullagh, P. and Nelder, J.A. (1989) Generalized Linear Models, 2nd   ed., London: Chapman & Hall.

24- Miller, 1990 - Miller, K. D. (1990). Correlation-based mechanisms of neural development. In Gluck, M. A. and Rumelhart, D. E., editors, Neuroscience and Connectionist Theory, pages 267--353. Erlbaum, Hillsdale NJ.

25- Minsky and Papert, 1969 - Minsky, M. and Papert, S. (1969). Perceptrons. MIT Press, Cambridge.

26- Moore et al, 94 - A.W.Moore, and M.S.Lee, Efficient algorithms for minimizing cross-validation error. In Proc. ML-94, Morgan Kaufmann, 1994.

27- Rosenblatt, 1962 - Rosenblatt, F. (1962). Principles of Neurodynamics. Spartan, New York.

28- Sahami, M.; Dumais, S.; Heckerman, D. and Horvitz, E. 1998. A Bayesian Approach to Filtering Junk E-mail. In Learning for Text Categorization—Papers from the AAAI Workshop (Madison, Wisconsin, 1998), AAAI Technical Report WS-98-05, pp. 55-62.

29- Sebastiani, F. 2002. Machine Learning in Automatic Text Categorization. ACM Computing Surveys (CSUR), 34(1): pp. 1-47.

30- Shopos Inc. 2010 Spam World Report

31- Skalak, 94- D.B.Skalak, Prototype and feature selection by sampling and random mutation hill climbing algorithms. In Proc. ML-94, Morgan Kaufmann, 1994.

32- Spertus, E. 1997. Smokey: Automatic Recognition of Hostile Messages. In Proceedings of the 14th National Conference on AI and the 9th Conference on Innovative Applications of AI (Providence, Rhode Island, 1997), pp. 1058-1065.

33- Weiss, A. 2003. Ending Spam's Free Ride. netWorker, 7(2): pp. 18-24.

34- W. Fuertes, M. Almache and J. Ruiz, "Clasificador de E-mails Anti-Spam utilizando un Perceptrón Multicapa", Revista de Ciencia y Tecnología de la Escuela Politécnica del Ejército, Volumen 2. Sangolquí, Ecuador, Mayo 2009.

11 - Software References

1- www.sinespam.com (professional project)

2- The "bpmaster" software was created from the PDP software.
The original PDP software can be found here:

http://www.stanford.edu/group/pdplab/resources.html#originalpdp.

The original PDP handbook can be found here:

http://www.stanford.edu/group/pdplab/originalpdphandbook/.

In the Appendix B there is a list of the original commands with their descriptions. Note, however, that it does not include some important commands, such as "set/ envrm", "set/ ensayo", "ensayo" or "backsel".

The current PDP software can be found here:

http://www.stanford.edu/group/pdplab/resources.html.

Annex I – Programs to obtain "the corpus"

Feature selection programs description

Clamav.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CLAMAV","/home/www/dataminig/clamav.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {
               echo "$file\n";

               $resultado=exec(CLAMAV.DIRECTORIO.$file);

               echo $resultado ;

          $sql="insert into data (filename,att_virus)
values('$file','$resultado')";
          $resultado=mysql_query($sql, $Bd );

          }//Endif

}//EndWhile
closedir($fdir);



?>
```

Spamassasin.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CRM","/home/www/dataminig/spamassassin.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);
```

```php
while ($file = readdir($fdir))
{
    if ($file !='.')
        if ($file !='..')
        {
            echo "$file\n";

            $resultado=exec(CRM.DIRECTORIO.$file);

            echo "$resultado\n";

        $sql="update data set att_spamassasin='$resultado'
where filename='$file'";
        $resultado=mysql_query($sql, $Bd );

        }//Endif

}//EndWhile
closedir($fdir);


?>
```

Contaracentos.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");

define("CADENAS","áéíóúÁÉÍÓÚñÑ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
    if ($file !='.')
        if ($file !='..')
        {
            echo "Procesando $file\n";

            $resultado=CuentaCaracteres(DIRECTORIO.$file);

            echo "RETORNO $resultado\n";

        $sql="update data set att_spanish='$resultado' where
filename='$file'";
        $resultado=mysql_query($sql, $Bd );

        }//Endif
```

```php
}//EndWhile
closedir($fdir);


function CuentaCaracteres($file)
{

//Leemos
$fp=fopen($file,"r");
$txt=fread($fp,filesize($file));
fclose($fp);

//Procesamos en busca de string en CONSTATE CADENAS
$strings=CADENAS;

//Numero de caracteres ha buscar
$reg=strlen($strings);

for ($i=0;$i<$reg;$i++)
{

     //Buscamos caracter
     $registros=explode($strings[$i],$txt);

     $TOTAL=$TOTAL+count($registros);

}//EndFor


//Devolvemos numero registros
return $TOTAL-$reg;

}//EndFunction

?>
```

Contarcaracteresraros.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");

define("CADENAS","¡!·#$€%~&/()=¿?*^{}[]°\@><¿'´+-_¨,");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {
```

```php
                echo "Procesando $file\n";

                $resultado=CuentaCaracteres(DIRECTORIO.$file);

                echo "RETORNO $resultado\n";

            $sql="update data set att_char_extend='$resultado'
where filename='$file'";
            $resultado=mysql_query($sql, $Bd );

            }//Endif


}//EndWhile
closedir($fdir);


function CuentaCaracteres($file)
{

//Leemos
$fp=fopen($file,"r");
$txt=fread($fp,filesize($file));
fclose($fp);

//Procesamos en busca de string en CONSTATE CADENAS
$strings=CADENAS;

//Numero de caracteres ha buscar
$reg=strlen($strings);

for ($i=0;$i<$reg;$i++)
{

     //Buscamos caracter
     $registros=explode($strings[$i],$txt);

     $TOTAL=$TOTAL+count($registros);

}//EndFor


//Devolvemos numero registros
return $TOTAL-$reg;

}//EndFunction

?>
```

Contarstringnospanish.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
```

```php
define("CMD","/home/www/dataminig/contar_string_nospanish.sh ");


//String ha buscar
define("CADENA","kqxzywv");

//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=ProcesaMensaje(DIRECTORIO.$file);

          echo "RESULTADO = $resultado \n";

          $sql="update data set att_char_nospanish='$resultado'
where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);

//Normalizamos fichero
function ProcesaMensaje($file)
{
echo "-----\n";
echo "Procesando $file\n";

$tmp="/tmp/nospanish/";
exec("rm -rf $tmp");
mkdir($tmp);
copy($file,$tmp."msg.txt");

//Normalizamos
exec(CMD." $tmp msg.txt");

//Procesamos SOLO ficheros txt
$fdir=opendir($tmp);
while ($file = readdir($fdir))
{
     if(eregi("textfile",$file))
     {
          echo "subfile $file \n";
          $nospanish=$nospanish+CuentaNospanish($tmp.$file);
     }

}//EndWhile

return $nospanish;
```

```php
}//EndFunction


//Contamos palabras con caracteres nospanish
function CuentaNospanish($file)
{
$cadena=CADENA;

//Leemos datos
$fp=fopen($file,"r");
$txt=fread($fp,filesize($file));
fclose($fp);

//Descomponemos en palabras
$words=explode(" ",$txt);

$reg=strlen($cadena);

//Procesamos cada palabra si hay algun letra
//kqxzywv
while (list($key, $value) = each($words))
{
     $KK=0;              //Reseteamos

     //quitamos . ya que evita filtrado
     $chkword=trim(str_replace(".","",$value));

     for ($i=0;$i<$reg;$i++)
     {
          if(eregi($cadena[$i],$value))
          {
               $KK++;
          }

     }//EndFor

//RESET
if(eregi("http",$value))
     $kk=0;
if(eregi("@",$value))
     $kk=0;


     if($KK>1)
     {
          echo "ENCONTRADO =========== $value \n";
          $TOTAL++;
     }

}//EndWhile

return $TOTAL;

}//EndFunction
```

```
?>
```

Contarstringsnowords.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/contar_string_nospanish.sh ");

//Longitud String ha buscar
define("LSTRING","10");

//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=ProcesaMensaje(DIRECTORIO.$file);

          echo "RESULTADO = $resultado \n";

          $sql="update data set att_long_words='$resultado'
where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);

//Normalizamos fichero
function ProcesaMensaje($file)
{
echo "-----\n";
echo "Procesando $file\n";

$tmp="/tmp/nospanish/";
exec("rm -rf $tmp");
mkdir($tmp);
copy($file,$tmp."msg.txt");

//Normalizamos
exec(CMD." $tmp msg.txt");

//Procesamos SOLO ficheros txt
$fdir=opendir($tmp);
while ($file = readdir($fdir))
{
     if(eregi("textfile",$file))
```

```php
    {
            echo "subfile $file \n";
            $longword=$longword+CuentaLogWord($tmp.$file);
    }
}//EndWhile


return $longword;


}//EndFunction



//Contamos palabras con LOG WORD
function CuentaLogWord($file)
{

//Leemos datos
$fp=fopen($file,"r");
$txt=fread($fp,filesize($file));
fclose($fp);

//Descomponemos en palabras
$words=explode(" ",$txt);


$reg=strlen($cadena);

//Procesamos cada palabra mirando su longitud

while (list($key, $value) = each($words))
{
//    echo strlen($value)."  =  ".$value."\n";

    //RESET
    if(eregi("http",$value))
        break;
    if(eregi("@",$value))
        break;

    if((strlen($value)>LSTRING))
        $KK++;

}//EndWhile

return $KK;

}//EndFunction


?>
```

Contarstringallupercase.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/contar_string_nospanish.sh ");
```

```php
//Longitud String ha buscar
define("LSTRING","10");

//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
    if ($file !='.')
        if ($file !='..')
        {

        $resultado=ProcesaMensaje(DIRECTORIO.$file);

        echo "RESULTADO = $resultado \n";

        $sql="update data set
att_char_alluppercase='$resultado' where filename='$file'";
        $resultado=mysql_query($sql, $Bd );

//exit;
        }//Endif

}//EndWhile
closedir($fdir);

//Normalizamos fichero
function ProcesaMensaje($file)
{
echo "-----\n";
echo "Procesando $file\n";

$tmp="/tmp/nospanish/";
exec("rm -rf $tmp");
mkdir($tmp);
copy($file,$tmp."msg.txt");

//Normalizamos
exec(CMD." $tmp msg.txt");

//Procesamos SOLO ficheros txt
$fdir=opendir($tmp);
while ($file = readdir($fdir))
{
    if(eregi("textfile",$file))
    {
        echo "subfile $file \n";
        $longword=$longword+CuentaAllUpper($tmp.$file);
    }
}//EndWhile

return $longword;

}//EndFunction
```

```php
//Contamos palabras con todas mayusculas
function CuentaAllUpper($file)
{

//Leemos datos
$fp=fopen($file,"r");
$txt=fread($fp,filesize($file));
fclose($fp);

//Descomponemos en palabras
$words=explode(" ",$txt);

$reg=strlen($cadena);

//Procesamos cada palabra comparando la misma en UpperCase
while (list($key, $value) = each($words))
{

     //Quitamos espacios y palabras menores de 2 letras
     $value=trim($value);
     if(strlen($value)<2)
          break;

     $testword=strtoupper($value);

     if($value==$testword)
     {
          echo $value."\n";
          $KK++;
     }//EndIf

}//EndWhile

return $KK;

}//EndFunction


?>
```

Contastringhref.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");

define("STRING","href");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
```

```php
        if ($file !='.')
            if ($file !='..')
            {
                    echo "Procesando $file\n";

                    $resultado=CuentaString(DIRECTORIO.$file);

                    echo "RETORNO $resultado\n";

            $sql="update data set att_body_href='$resultado'
where filename='$file'";
            $resultado=mysql_query($sql, $Bd );

//exit;
            }//Endif


}//EndWhile
closedir($fdir);


function CuentaString($file)
{

//Leemos
$fp=fopen($file,"r");
$txt=fread($fp,filesize($file));
fclose($fp);

//Procesamos en busca de string en CONSTATE CADENAS

    //Buscamos caracter
    $registros=explode(STRING,$txt);

    $TOTAL=$TOTAL+count($registros);



//Devolvemos numero registros
return $TOTAL-1;

}//EndFunction

?>
```

dcc.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/dcc.sh ");


unlink("/tmp/alldcc.log");
```

```php
//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {
               echo "$file\n";

               $resultado=exec(CMD.DIRECTORIO.$file);

               echo "$resultado\n";

          $sql="update data set att_dcc='$resultado' where
filename='$file'";
          $resultado=mysql_query($sql, $Bd );

          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Razor.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/razor.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {
               echo "$file\n";

               $resultado=exec(CMD.DIRECTORIO.$file);

               echo "$resultado\n";

          $sql="update data set att_razor='$resultado' where
filename='$file'";
          $resultado=mysql_query($sql, $Bd );

          }//Endif
```

```
}//EndWhile
closedir($fdir);



?>
```

Pyzor.php

```
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/pyzor.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
           if ($file !='..')
           {
                echo "$file\n";

                $resultado=exec(CMD.DIRECTORIO.$file);

                echo "$resultado\n";

           $sql="update data set att_pyzor='$resultado' where
filename='$file'";
           $resultado=mysql_query($sql, $Bd );

           }//Endif

}//EndWhile
closedir($fdir);



?>
```

Bodyhref.php

```
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/body_href_extend.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
```

```
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);

          echo "RESULTADO $file = $resultado \n";

          $sql="update data set
att_body_href_extend_char='$resultado' where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Bodyimghref.php

```
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/body_img_href.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);

          echo "RESULTADO $file = $resultado \n";

          $sql="update data set att_body_href_img='$resultado'
where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Bodycolor.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/body_color_white.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);

          echo "RESULTADO $file = $resultado \n";

          $sql="update data set att_body_color='$resultado'
where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Fromurl.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/from_with_url.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);
```

```php
            echo "RESULTADO $file = $resultado \n";

            $sql="update data set att_count_url='$resultado'
where filename='$file'";
            $resultado=mysql_query($sql, $Bd );

//exit;
            }//Endif

}//EndWhile
closedir($fdir);


?>
```

Htmlalternative.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/html_alternative.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
           if ($file !='..')
           {

           $resultado=exec(CMD.DIRECTORIO.$file);

           echo "RESULTADO = $resultado \n";

           $sql="update data set att_header_html='$resultado'
where filename='$file'";
           $resultado=mysql_query($sql, $Bd );

//exit;
           }//Endif

}//EndWhile
closedir($fdir);


?>
```

Dkim.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
```

```php
define("CMD","/home/www/dataminig/dkim.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);

          echo "RESULTADO = $resultado \n";

          $sql="update data set att_dkim='$resultado' where
filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Contarcharnumber3_char.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/contar_char3_number.sh ");

//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=ProcesaMensaje(DIRECTORIO.$file);

          echo "RESULTADO = $resultado \n";

          $sql="update data set att_char3_number='$resultado'
where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
```

```php
            }//Endif

    }//EndWhile
    closedir($fdir);

//Normalizamos fichero
function ProcesaMensaje($file)
{
echo "-----\n";
echo "Procesando $file\n";

$tmp="/tmp/contar_char3/";
exec("rm -rf $tmp");
mkdir($tmp);
copy($file,$tmp."msg.txt");

//Normalizamos
exec(CMD." $tmp msg.txt");

//Procesamos SOLO ficheros txt
$fdir=opendir($tmp);
while ($file = readdir($fdir))
{
    if(eregi("textfile",$file))
    {
            echo "subfile $file \n";

            //Leemos fichero y procesamos lineas/palabras
            //Devolviendo palabras con numeros
            $tword=$tword+ProcesaFichero($tmp.$file);

    }//EndIf

}//EndWhile

return $tword;

}//EndFunction


//Leemos fichero y procesamos lineas/palabras
function ProcesaFichero($file)
{

$lineas=file($file);

//Recorremos lineas
while (list($key, $value) = each($lineas))
{
    #Buscamos numeros en lineas
    if(ereg("[0123456789]",$value))
    {
            $word=explode(" ",$value);

            //Contamos palabras con numeros
            while (list($bkey, $bvalue) = each($word))
```

```
                {
                        if(ereg("[0123456789]",$bvalue))
                                $nword++;
                }//EndWhile

        }//EndIf

}//EndWhile


return $nword;

}//EndFunction

?>
```

Fromconsonantes.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/from_consonantes.sh ");


//cuentas consonantes en el remitente

//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);

          $resultado=str_replace("a","",$resultado);
          $resultado=str_replace("e","",$resultado);
          $resultado=str_replace("i","",$resultado);
          $resultado=str_replace("o","",$resultado);
          $resultado=str_replace("u","",$resultado);

          $nconsonantes=strlen($resultado);

          echo "RESULTADO $file =  $nconsonantes $resultado
\n";

          $sql="update data set att_char_from_consonante
sender='$nconsonantes' where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif
```

```
}//EndWhile
closedir($fdir);


?>
```

Contarcharvocal.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/contar_char3_vocal.sh ");

//Contamos palabras con mas de tres vocales

//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
           if ($file !='..')
           {

           $resultado=ProcesaMensaje(DIRECTORIO.$file);

           echo "RESULTADO = $resultado \n";

           $sql="update data set att_vocal='$resultado' where
filename='$file'";
           $resultado=mysql_query($sql, $Bd );

//exit;
           }//Endif

}//EndWhile
closedir($fdir);



//Normalizamos fichero y buscamos
function ProcesaMensaje($file)
{
echo "-----\n";
echo "Procesando $file\n";

$tmp="/tmp/contar_char3/";
exec("rm -rf $tmp");
mkdir($tmp);
copy($file,$tmp."msg.txt");

//Normalizamos
exec(CMD." $tmp msg.txt");

//Procesamos SOLO ficheros txt
```

```php
$fdir=opendir($tmp);
while ($file = readdir($fdir))
{
      if(eregi("textfile",$file))
      {
            echo "subfile $file \n";
            $valor=exec(CMD.$tmp." ".$file);

            $CHARVOCAL=$CHARVOCAL+$valor;
      }

}//EndWhile

return $CHARVOCAL;

}//EndFunction



?>
```

Spffrom.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/spf_from.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
      if ($file !='.')
            if ($file !='..')
            {

            $resultado=exec(CMD.DIRECTORIO.$file);

            echo "RESULTADO $file = $resultado \n";

            $sql="update data set att_spf='$resultado' where
filename='$file'";
            $resultado=mysql_query($sql, $Bd );

//exit;
            }//Endif

}//EndWhile
closedir($fdir);
```

```
?>
```

Mxfrom.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/mx_from.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);

          echo "RESULTADO $file = $resultado \n";

          $sql="update data set att_mx_sender='$resultado'
where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Contarcontenttype.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/contar_contentype.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);
```

```php
            echo "RESULTADO = $resultado \n";

//          $sql="update data set att_header_html='$resultado'
where filename='$file'";
//          $resultado=mysql_query($sql, $Bd );

//exit;
            }//Endif

}//EndWhile
closedir($fdir);


?>
```

Contarheaderpriority.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CMD","/home/www/dataminig/contar_header_priority.sh ");

//borramos fichero de log de cabeceras encontradas
unlink("/tmp/priority.log");

//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {

          $resultado=exec(CMD.DIRECTORIO.$file);

          echo "RESULTADO = $resultado \n";

          $sql="update data set
att_header_priority='$resultado' where filename='$file'";
          $resultado=mysql_query($sql, $Bd );

//exit;
          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Crm114.php

```php
<?
```

```php
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CRM","/home/www/dataminig/crm114.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
            {
                echo "$file\n";

                $resultado=exec(CRM.DIRECTORIO.$file);

                echo "$resultado\n";

          $sql="update data set att_crm114='$resultado' where
filename='$file'";
          $resultado=mysql_query($sql, $Bd );

          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Contarstringraros.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");

define("CMD","/home/www/dataminig/contar_string_rcpt.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
            {
                echo "-----\n";
                echo "Procesando $file\n";

                $resultado=exec(CMD.DIRECTORIO.$file);
```

```php
                //Quitamos From:
                $txt=str_replace("From: ","",$resultado);

                $resultado=CuentaConsonantes($txt);

                echo "RETORNO $resultado\n";

            $sql="update data set
att_char_header_consonante='$resultado' where filename='$file'";
            $resultado=mysql_query($sql, $Bd );


            }//Endif

}//EndWhile
closedir($fdir);



//Contamos palabras y si hay alguna sin vocales
function CuentaConsonantes($string)
{

//Descomponemos en palabras
$words=explode(" ",$string);

//Procesamos cada palabra si hay algun vocal
while (list($key, $value) = each($words))
{
     echo "$key: $value\n";

     if(eregi("a",$value))
          break;

     if(eregi("e",$value))
          break;

     if(eregi("i",$value))
          break;

     if(eregi("o",$value))
          break;

     if(eregi("u",$value))
          break;

     echo "ENCONTRADO  --- $value -- \n";

     $TOTAL++;

}//EndWhile

return $TOTAL;

}//EndFunction
```

```
?>
```

Virus.php

```php
<?
include("dbconfig.php");

//Demo desarrollos

define("DIRECTORIO","/home/backup/TESTALL/");
define("CLAMAV","/home/www/dataminig/clamav.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
     if ($file !='.')
          if ($file !='..')
          {
               echo "$file\n";

//               $resultado=exec(CLAMAV.DIRECTORIO.$file);

               echo $resultado ;

//          $sql="insert into data (filename,att_virus)
values('$file','$resultado')";
//          $resultado=mysql_query($sql, $Bd );

          }//Endif

}//EndWhile
closedir($fdir);


?>
```

Dbconfig.php

```php
<?
//Conexion Server MySql
$Bd=mysql_connect("213.XXX.XXX.XXX","datamining" ,"datamining"
);

//Conexion BBDD
$result = mysql_select_db("corpus", $Bd );


?>
```

Sanesecurity.php

```php
<?
include("dbconfig.php");

define("DIRECTORIO","/home/backup/TESTALL/");
define("CLAMAV","/home/www/dataminig/sanesecurity.sh ");


//Leemos ficheros
$fdir=opendir(DIRECTORIO);

while ($file = readdir($fdir))
{
    if ($file !='.')
        if ($file !='..')
        {

            $resultado=exec(CLAMAV.DIRECTORIO.$file);

            echo "$resultado $file\n";

        if($resultado==1)     //SPAM
            $sql="update data set
att_sane_spam='1',att_sane_virus='0' where filename='$file'";

        if($resultado==2)     //VIRUS
            $sql="update data set
att_sane_virus='1',att_sane_spam='0' where filename='$file'";

        if($resultado==0)     //Nada
            $sql="update data set
att_sane_virus='0',att_sane_spam='0' where filename='$file'";


        echo $sql."\n";
        $resultado=mysql_query($sql, $Bd );

        }//Endif

}//EndWhile
closedir($fdir);



?>
```

Annex II – Features Test Classifications

You may see the feature test classification on the CD-ROM attached to this document

Annex III – DataBase Definition using MySQL

Campo	Tipo	Collation	Atributos	Nulo	Predeterminado	Extra	Acción					
☐	id	int(11)		Sí	NULL	auto_increment						
☐	att_spamassasin	int(11)		Sí	NULL							
☐	att_spanish	int(11)		Sí	NULL							
☐	att_vocal	int(11)		Sí	NULL							
☐	att_body_normalize	int(11)		Sí	NULL							
☐	att_char_extend	int(11)		Sí	NULL							
☐	att_char3_number	int(11)		Sí	NULL							
☐	att_char_header_consonante	int(11)		Sí	NULL							
☐	att_char_nospanish	int(11)		Sí	NULL							
☐	att_long_words	int(11)		Sí	NULL							
☐	att_char_alluppercase	int(11)		Sí	NULL							

☐	att_char_repeat	int(11)		Sí	*NULL*		
☐	att_header_priority	int(11)		Sí	*NULL*		
☐	att_header_html	int(11)		Sí	*NULL*		
☐	att_char_body_consonante	int(11)		Sí	*NULL*		
☐	att_body_href	int(11)		Sí	*NULL*		
☐	att_mx_sender	int(11)		Sí	*NULL*		
☐	att_body_href_img	int(11)		Sí	*NULL*		
☐	att_body_color	int(11)		Sí	*NULL*		
☐	att_body_href_extend_char	int(11)		Sí	*NULL*		
☐	att_dkim	int(11)		Sí	*NULL*		
☐	att_sender_extend_char	int(11)		Sí	*NULL*		
☐	att_crm114	int(11)		Sí	*NULL*		
☐	att_max_uppercase	int(11)		Sí	*NULL*		
☐	att_long_uppercase	int(11)		Sí	*NULL*		

☐	att_rbl	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_count_url	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_permissive_words	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_spf	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_virus	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	filename	char(25)	latin1_swedish_ci	Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_sane_spam	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_sane_virus	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_resultado	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_razor	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_pyzor	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_dcc	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T
☐	att_char_from_consonante	int(11)		Sí	*NULL*			✎	✗	🔑	✎	U	T

Annex IV – Feature Classification maximum accuracy

FINAL MAXIMUM CLASSIFICATION USING 13 FEATURES

****

******** RESULTADOS MEDIOS DE LOS ULTIMOS 100 ENSAYOS *******

****

--------- RESULTADOS MEDIOS MAXIMOS POR ERROR MEDIO --------

MAXIMOS EN EL CONJUNTO DE TRAIN

 Numero de resultados obtenidos: 100

 epoch: 400.000000 +/-0.00

 TRAIN   tss: 41.04 +/-2.58 (0.03) aciertos: 1273.34/1320.00 (96.47% +/-0.36%)

   Porcentajes medios de aciertos de cada clase: 97.87% 95.06%

 VALIDATION tss: 15.18 +/-1.97 (0.03) aciertos: 422.61/440.00 (96.05% +/-0.64%)

   Porcentajes medios de aciertos de cada clase: 97.64% 94.47%

 TEST   tss: 15.18 +/-1.97 (0.03) aciertos: 422.65/440.00 (96.06% +/-0.64%)

   Porcentajes medios de aciertos de cada clase: 97.63% 94.49%

 MAXIMOS EN EL CONJUNTO DE VALIDATION

 Numero de resultados obtenidos: 100

 epoch: 388.630000 +/-42.54

 TRAIN   tss: 41.26 +/-2.54 (0.03) aciertos: 1273.01/1320.00 (96.44% +/-0.35%)

   Porcentajes medios de aciertos de cada clase: 97.86% 95.02%

 VALIDATION tss: 15.16 +/-1.96 (0.03) aciertos: 422.68/440.00 (96.06% +/-0.62%)

   Porcentajes medios de aciertos de cada clase: 97.64% 94.50%

 TEST   tss: 15.23 +/-1.95 (0.03) aciertos: 422.66/440.00 (96.06% +/-0.65%)

   Porcentajes medios de aciertos de cada clase: 97.65% 94.48%

MAXIMOS EN EL CONJUNTO DE TEST

Numero de resultados obtenidos: 100

epoch: 388.610000 +/-42.74

TRAIN   tss: 41.26 +/-2.54 (0.03) aciertos: 1273.02/1320.00 (96.44% +/-0.35%)

  Porcentajes medios de aciertos de cada clase: 97.86% 95.02%

VALIDATION tss: 15.23 +/-1.95 (0.03) aciertos: 422.63/440.00 (96.05% +/-0.65%)

  Porcentajes medios de aciertos de cada clase: 97.65% 94.46%

TEST   tss: 15.17 +/-1.96 (0.03) aciertos: 422.70/440.00 (96.07% +/-0.62%)

  Porcentajes medios de aciertos de cada clase: 97.63% 94.52%

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Tiempos de ejecucion:

  Wed Sep 01 22:27:40 2010

  Thu Sep 02 00:35:31 2010

  0 dias / 2 horas / 7 minutos / 51.00 segundos

Annex V – Corresponding Classification using BPMaster

This is the reference of the files on the CD-ROM included with this thesis work.
For example, the attribute for the file = att_spanish → is → spam2.pat

Tipo	5 CV Fold	10 CV Fold
id	Not	Not
att_spamassasin	spam1	spam01
att_spanish	spam2	spam02
att_vocal	spam15	spam015
att_body_normalize	Not	Not

att_char_extend	spam3	spam03
att_char3_number	spam16	spam016
att_char_header_consonante	spam17	spam017
att_char_nospanish	spam4	spam04
att_long_words	spam5	spam05
att_char_alluppercase	spam6	spam06
att_char_repeat	spam18	spam018
att_header_priority	spam7	spam07
att_header_html	spam19	spam019
att_char_body_consonante	Not	Not
att_body_href	spam8	spam08
att_mx_sender	spam20	spam020
att_body_href_img	spam21	spam021
att_body_color	spam22	spam022
att_body_href_extend_char	spam23	spam023
att_dkim	spam24	spam024
att_sender_extend_char	Not	Not
att_crm114	spam9	spam09
att_max_uppercase	Not	Not
att_long_uppercase	Not	Not
att_rbl	Not	Not
att_count_url	spam25	spam025
att_permissive_words	Not	Not
att_spf	spam26	spam026
att_virus	spam10	spam010
filename	Not	Not
att_sane_spam	spam11	spam011
att_sane_virus	spam12	spam012
att_resultado	Not	Not
att_razor	spam13	spam013
att_pyzor	spam27	spam027
att_dcc	spam14	spam014
att_char_from_consonante	spam28	spam028

Annex VI – Corpus – Dataset

You may see the corpus dataset on the CD-ROM attached this document file called corpustotal.pat