Mittuniversitetet
MID SWEDEN UNIVERSITY

Master Thesis

IMPROVING NAIVE BAYESIAN SPAM FILTERING

Jon Kågström

Mid Sweden University
Department for Information Technology and Media
Spring 2005

Abstract

Spam or unsolicited e-mail has become a major problem for companies and private users. This thesis explores the problems associated with spam and some different approaches attempting to deal with it. The most appealing methods are those that are easy to maintain and prove to have a satisfactory performance. Statistical classifiers are such a group of methods as their ability to filter spam is based upon the previous knowledge gathered through collected and classified e-mails. A learning algorithm which uses the Naive Bayesian classifier has shown promising results in separating spam from legitimate mail. Tokenization, probability estimation and feature selection are processes performed prior to classification and all have a significant influence upon the performance of spam filtering. The main objective of this work is to examine and empirically test the currently known techniques used for each of these processes and to investigate the possibilities for improving the classifier performance. Firstly, how a filter and wrapper approach can be used to find tokenization delimiter subsets that improve classification is shown. After this, seven probability estimators are tested and compared in order to demonstrate which of them ameliorate the performance. Finally a survey of commonly used methods for the feature selection process is performed and recommendations for their use are presented.

Acknowledgments

I would like to thank my supervisor Iskra Popova for guiding me in how to write a thesis. She has helped me to improve the structure and pointed out where the content needed enrichment. For me it has been a learning process. I also want to thank everyone who makes an effort to fight spam, and there are a lot of you!

Glossary

classifier	A person or machine that is sorting out the constituents of a substance.
clique	A maximum complete subgraph of a graph.
complete graph	All vertices are connected with each other.
corpus	A collection of natural language text used for accumulating statistics. More specifically in this thesis a corpus is a map between a word (token) and its frequency.
degrees of freedom	Describe the number of values that are free to vary in a statistical calculation.
delimiter	A character that marks the beginning or end of a unit of data.
entropy	A measure of the disorder that exists in a system.
feature	A prominent part or characteristic.
inducer	An inducer is a machine learning algorithm that produces a classifier that, in turn, assigns a class to each instance.
misclassification	An acctual spam classified as good, or an acctual good classified as spam.
monotonicity	The function f is monotone if, whenever $x \le y$, then $f(x) \le f(y)$. Stated differently, a monotone function is one that *preserves the order*.
mutually exclusive	Describing two events, conditions, or variables which cannot occur at once.
n-gram	n features are considered at a time.
null hypothesis	Predicts that two distributions are the same.
probability distribution	A list of the probabilities associated with each of its values. Throughout this work discreet probability distributions are used and they are non-continuous. The probability mass function is denoted by $p(x_i)$ where x_i is a random discrete variable.
sample space	The set of all possible outcomes of an experiment.
significance level	Is the decision criterion for accepting or rejecting the null hypothesis.
space	Character used to separate words.
spam	Unsolicited usually commercial e-mail sent to a large number of addresses.
token	A distinguishing characteristic (feature).
transitivity	In mathematics, a binary relation R over a set X is transitive if it holds for all a, b, and c in X, that if a is related to b and b is related to c, then a is related to c.
unigram	Only one feature is considered at a time. See also n-

	gram.
white-space	The characters, space, tab, line-feed and other characters that leaves an empty space.
χ^2 **-distribution**	The shape alters as the degrees of freedom change. The area under the curve will grow as the degrees of freedom increase making it more symmetrical. With more than 30 degrees of freedom it approximates the normal distribution.

Abbreviations

ASCII	American Standard Code of Information Interchange
BNS	Bi-normal separation
ELE	Expected Likelihood Estimate
HTML	Hyper Text Markup Language
IP	Internet Protocol
ISP	Internet Service Provider
KL	Kullback-Liebler
kNN	k-Nearest Neighbor
L-S	Ling-Spam
MLE	Maximum Likelihood Estimate
NB	Naive Bayesian
NNet	Neural Network
SA	SpamAssassin
SBS	Sequential Backward Selection
SBFS	Sequential Backward Floating Selection
SFS	Sequential Forward Selection
SFFS	Sequential Forward Floating Selection
SMTP	Simple Mail Transfer Protocol
SVM	Support Vector Machine
XML	Extensible Markup Language

Notation

α	Significance level.
χ^2	Chi square statistics.
λ	Weight of each good message.
A	Alphabet.
B	Number of bins (distinct items).
C	Class vector.
D	Delimiter set
EW_{Acc}	Extended Weighted Accuracy.
F	Feature vector.
IG	Information Gain.
J	Fitness function.
KL	Kullback-Liebler divergence.
L	Label vector.
M	Message corpus.
n_{gg}	Number of good messages classified as good.
n_{gs}	Number of good messages classified as spam.
n_{ss}	Number of spam messages classified as spam.
n_{sg}	Number of spam messages classified as good.
N	Total number of training instances.
N_j	Frequency of frequency of items seen j times.
p	Precision
p_{MLE}	Maximum likelihood estimator.
p_{Abs}	Absolute estimator.
p_{Lap}	Laplace estimator.
p_{ELE}	Expected likelihood estimator.
p_{Lid}	Lidstone estimator.
p_{WB}	Witten Bell estimator.
p_{GT}	Good Turing estimator.
p_{Rob}	Bayesian estimator.
$p-value$	Calculated probability in classical statistics.
P	Probability.
PR	Probability ratio.
Q	Production of a tokenizer.
r	Recall.
R	Rank.
t	Threshold value for spam cutoff.
T	Tokenizer.
W_{Acc}	Weighted accuracy.

X , Y	Events.

Table of Contents

List of Figures

1 Introduction

1.1 The problem of spam

Internet has opened new channels of communication; enabling an e-mail to be sent to a relative thousand of kilometers away. This medium of communication opens doors for virtually free mass e-mailing, reaching out to hundred of thousands users within seconds. However, this freedom of communication can be misused. In the last couple of years spam has become a phenomenon that threatens the viability of communication via e-mail.

It is difficult to develop an accurate and useful definition of spam, although every e-mail user will quickly recognize spam messages. Merriam-Webster Online Dictionary[1] defines spam as "unsolicited usually commercial e-mail sent to a large number of addresses". Some other than commercial purposes of spam are to express political or religious opinions, deceive the target audience with promises of fortune, spread meaningless chain letters and infect the receivers' computer with viruses. Even though one can argue that what is spam for one person can be an interesting mail message for another, most people agree that spam is a public frustration.

Spam has become a serious problem because in the short term it is usually economically beneficial to the sender. The low cost of e-mail as a communication medium virtually guaranties profits. Even if a very small percentage of people respond to the spam advertising message by buying the product, this can be worth the money and the time spent for sending bulk e-mails. Commercial spammers are often represented by people or companies that have no reputation to lose. Because of technological obstacles with e-mail infrastructure, it is difficult and time-consuming to trace the individual or the group responsible for sending spam. Spammers make it even more difficult by hiding or forging the origin of their messages. Even if they are traced, the decentralized architecture of the Internet with no central authority makes it hard to take legal actions against spammers.

Spam has increased steadily over the last years, according to Brightmail[2]. At present, March 2004, 62% of all emails on the internet are spam compared to 45% a year ago. The major problem concerning spam is that it is the receiver who is paying for the spam in terms of their time, bandwidth and disk space. This can be very costly even for a small company with only 20 employees who each receive 20 spam e-mails a day. If it takes 5 seconds to classify and remove a spam, then the company will spend about half an hour every day to separate spam from legitimate e-mail. The statistics shows that 20 spam messages per day is a very low number for a company that is susceptible to spam. There are other problems associated with spam. Messages can have content that is offensive to people and might cause general psychological annoyance, a large amount of spam messages can crash unprotected mail servers, legitimate personal e-mails can be easily lost and more.

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

[1] Merriam-Webster Online Dictionary, http://www.m-w.com/, 2004-03-12

[2] Brightmail, http://brightmail.com/, 2004-03-12

1.2 Research objectives

There are many different approaches available at present attempting to solve the spam issue. One of the most promising methods for filtering spam with regards to performance and ease of implementation is that of statistical filters. These filters learn to distinguish (or classify) between spam and legitimate e-mail messages as they are being used. In addition, they automatically adapt as the content of spam messages changes.

The objective of this thesis is to explore the statistical filter called Naive Bayesian classifier and to investigate the possibilities for improving its performance. After dissecting the segments of its operation, this work focuses on three specific areas described below.

- Before a message can be classified as either spam or legitimate it is first split into tokens; this process is called tokenizing. As this text is being read, tokenizing into tokens (words) is actually taking place as space is being used as a delimiter. Similarly an e-mail message can be split into tokens using space or any other character as delimiter. The first objective of this work is to examine how the selection of delimiters affects the classifier's performance and to offer recommendations for choosing delimiters.

- The classification of some e-mail messages as spam is based upon the knowledge gathered from the statistics about tokens appearing in previous e-mail messages. When a message is to be classified; each token is looked up in the training data. For example, the token 'Viagra' may have appeared 5 times in previous spams and 0 times in previous legitimate e-mails. These are the frequencies of a token in the training data. From these frequencies it is possible to estimate the probability that a token is found in a spam or legitimate e-mail. The most straight forward technique is to divide the frequency by the total number of tokens previously seen. Higher frequencies give better probability estimates. But whenever a token is either not present in any of the previous messages or it has a low frequency, there are better ways of estimating its probability. Our second objective is to examine how different probability estimators affect the spam classification performance.

- A feature is a characteristic of an object. For example in image recognition a feature could be a color and in the case of classifying e-mails it is a token or a word. E-mail messages are written using natural languages which contain thousands of distinct words. The number of words is the dimensionality of the message. The primary purpose of feature selection is to reduce the dimensionality in order to increase the speed of the computation. Our third objective was to conduct comparative analyses between three commonly used feature selection methods.

1.3 Thesis Outline

The thesis is structured in seven chapters. Chapter two discusses the method used. Chapter three briefly describes currently developed techniques to eliminate spam aiming to show that all existing schemes are not fully developed and do not offer complete spam elimination. It also emphasizes that statistical filters have, so far, proved to be the most successful method in dealing with spam.

Chapter four gives an overview of statistical classifiers and provides the reader with some necessary basic mathematical understanding for this work. Chapter five presents a general conceptualized model of a Naive Bayesian spam filter and explains the theory used by the Naive Bayesian classifier. This chapter also contains the currently used techniques for the three phases performed prior to the classification of messages starting with an examination of how the selection of delimiters affects the classifier performance. Following this seven different approaches are presented for smoothing the probabilities for the training data and finally three different methods for selecting features are demonstrated. Chapter six is devoted to the experimental work performed. It contains three sections, each with its own conclusions and one for each experiment. In this way the results presented in the sections are summarized and the conclusions are derived. Chapter seven is a summarization of all the work performed.

2 Method

The methodology used throughout the thesis consisted of a theoretical study requiring a literature survey and practical work involving several experiments.

2.1 Literature Survey

Articles found on the Internet are the most commonly used research material for this work. Google and Citeseer[3] were frequently used to find articles of interest. The spam phenomenon is still in its infancy and it was therefore natural to use the Internet as the main source of information. The theory behind statistical filters is well established and a number of books on statistics served as primary literature in this area. Books on Formal Languages, Artificial Intelligence and Discrete Mathematics were often consulted throughout the work on this thesis.

2.2 Experimental work

The experimental work is supported by some theoretical background. The empirical results obtained were verified with the theoretical ones whenever they were available. To carry out the experiments a test environment in C++ was built. In order to avoid rebuilding the environment for different tests, experiments were defined in an XML file that is read at run-time. For example, the corpus to use, probability estimator and feature selection method are defined in the XML file.

[3] http://citeseer.ist.psu.edu/ is a database with articles.

3 Techniques to eliminate spam

There are several approaches which deal with spam. This section briefly summarizes some common methods to avoid spam and briefly describes the spam filtering techniques used at present.

3.1 Hiding the e-mail address

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

3.2 Pattern matching, whitelists and blacklists

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

3.3 Rule based filters

This is a popular content-based method deployed by spam filtering software such as SpamAssassin[5]. Rule-based filters apply a set of rules to every incoming email. If there is a match, the e-mail is assigned a score that indicates spaminess or non-spaminess. If the total score exceeds a threshold the e-mail is classified as spam. The rules are generally built up by regular expressions and they come with the software. The rule set must be updated regularly as spam changes, in order for the filtering of spam to be successful. Updates are retrieved via the Internet. The tests results from the comparison of anti-spam programs presented in Holden 2003 show that SpamAssasin finds about 80% of all spam, while statistical filters (discussed later) find close to 99% of all spam.

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

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

The advantage of rule-based filters is that they require no training to perform reasonably well. Rules are implemented by humans and they can be very complex. Before a newly written rule is ready for use, it requires extensive testing to make sure it only classifies spam as spam and not legitimate messages as spam. Another disadvantage of this technique is the need for frequent updates of the rules. Once the spammer finds the way to deceive the filter, the spam messages will get through all filters with the same set of rules.

3.4 Statistical filters

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

3.5 E-mail verification

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

3.6 Distributed blacklists of spam sources

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

[6] Bluebottle, http://www.bluebottle.com/

[7] ChoiceMail, http://www.digiportal.com/index.html

mail communications are with other serious non-listed businesses. Companies offering this service include MAPS[8], ORDB[9] and Spamcop[10].

3.7 Distributed blacklist of spam signatures

These blacklists work in a same manner to that described in 3.6. The difference is that these blacklists consist of spam message signatures instead of spam sources. When a user receives a spam, that user can report the message signature (typically a hash code of the e-mail) to the blacklist. In this way, one user will be able to warn all other users that a certain message is spam. To avoid non-spam being added to a distributed blacklist, many different users must have reported the same signature. Spammers have found an easy way to fool these filters; they simply add a random string to every spam. This will prevent the e-mail from being detected in the blacklist. However spam fighters attempt to overcome this problem by adapting their signature algorithms to allow some random noise. The advantage being that these kinds of filters rarely classify legitimate messages as spam. The greatest disadvantage is they are not able to recall much of the spam. Vipul's Razor[11] uses such a blacklist and states that it catches 60%-90% of all incoming spam. Another disadvantage is the time taken for the network lookup.

3.8 Money e-mail stamps

The idea of e-mail stamps is not new, having been discussed since 1992, but it is not until recently that major companies have considered using it to combat spam. The sender would have to pay a small fee for the stamp. This fee could be minor for legitimate e-mail senders, while it could destroy business for spammers that send millions of e-mails daily. There are two stamp types; money stamps and proof-of-work stamps (discussed later). GoodmailSystems[12] is developing a system for money stamps. The basic idea is to insert a unique encrypted id to the header of each sent e-mail. If the recipient ISP is also participating in the system, the id is sent to Goodmail where it is decrypted. Goodmail will now be able to identify and charge the sender of the e-mail. Today there are many issues requiring solutions before such a system can be deployed. Who receives the money? Where is tax paid? Who are allowed to sell stamps? Since this is a centralized solution, what about scalability? It would also be the end of many legitimate newsletters.

3.9 Proof-of-work e-mail stamps

At the beginning of 2004, Bill Gates, Microsoft's chairman, suggested that the spam problem could be solved within two years by adding a proof-of-work stamp to each e-mail. Camram[13] is a system that uses proof-of-work stamps. Instead of taking a micro fee from the sender, a cheat-proof mathematical puzzle is sent. The puzzle requires a certain amount of computational power to be

[8] Mail Abuse Prevention System LLC (MAPSSM), http://mailabuse.com/

[9] Open Relay DataBase (ORDB), http://ordb.org/

[10] Spamcop, http://www.spamcop.net/

[11] Vipul's Razor, http://razor.sourceforge.net/

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

[13] Camram, http://www.camram.org/

solved (matter of seconds). When a solution is found, it is sent back to the receiver and the e-mail is allowed to pass to the receiver. The puzzle Camram is using is called Hashcash[14].

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

3.10 Legal measures

In recent years many nations have introduced anti-spam laws, in December 2003, president George W. Bush signed the CAN-SPAM[15] act, the Controlling the Assault of Non-Solicited Pornography and Marketing Act. The law prohibits the use of forged header information in bulk commercial e-mail. It also requires spam to include opt-out instructions. Violations can result in fines of $250 per e-mail, capped at $6 million. In April 2004 the first four spammers were charged under the CAN-SPAM law. The trial is still on, but if the court manages to send out a strong message, this could deter some spammers. The European Union introduced an anti-spam law on the 31st of October 2003 called "The Directive on Privacy and Electronic Communications". This new law requires that companies gain consent before they send out commercial e-mails. Many argue that this law is toothless since most of the spam comes from the outside of EU. In the long-run legislation can be used to slowdown the spam flood to some extent, but it will require an international movement. Legislation will not be able to solve the spam problem by itself, at least not in the near future.

3.11 Conclusion

The most commonly used methods for eliminating spam were described in this chapter. Perhaps legislation is the best option in the long run. However, it requires a world wide effort and this process could be slow. Presently users need to protect themselves and for the moment statistical filters are the most promising method for this purpose. They have superior performance, can adapt automatically as spam changes and in many cases are computationally efficient.

[14] Hashcash, http://www.hashcash.org/

[15] Information about spam laws can be found here http://www.spamlaws.com/.

4 Statistical Classifiers

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

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

4.1 Features and classes

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

4.2 Text categorization

Text categorization is the problem involved in classifying text documents to a category or class. Text categorization is becoming more popular as the amount of digital textual information grows. The problem of classifying an e-mail message as spam or legitimate message can be considered as a text categorization problem. Another popular area of use is Web page categorization to hierarchical catalogues.

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

There are many statistical filters in the literature. An extensive study (Yang Y. and Liu X, 1998) compared several filters including Supported Vector Machines (SVM), k-Nearest-Neighbor (kNN), Neural Networks (NNet) and Naive Bayesian (NB). NB is the only generative algorithm from these four. A brief introduction to these classifiers follows.

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

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

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

Among the described classifiers the NB classifier is the simplest in terms of its ease of implementation. When compared to the others, it is also computationally efficient. Tests carried out by Yang Y. and Liu X, 1998 showed that NB underperformed the others. Another study (Androutsopoulos 2000a) shows that NB outperforms kNN. For this work NB is used as classifier not only for its simplicity and computational efficiency, but also because of a belief that with a good probability estimator and careful feature selection it does not necessarily under-perform discriminative methods.

In the rest of this section the basic elements of the theory relevant to NB will be clarified by using simple examples from everyday life. The detailed description of NB and its elements will follow in the next chapter.

4.3 Basics about Probability Theory

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

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

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

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

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

$$P(Y \mid X) = \frac{P(X \wedge Y)}{P(X)}. \tag{2}$$

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

$$P(Y \mid X) = \frac{P(X \wedge Y)}{P(X)} = \frac{P(X) \cdot P(Y)}{P(X)} = P(Y) \tag{3}$$

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

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

$$P(S \mid V) = \frac{P(V \wedge S)}{P(V)} = \frac{0.20}{0.60} = \frac{1}{3}$$

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

4.4 Bayes theorem

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

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

This simple formula has enormous practical importance in many applications. It is often easier to calculate the probabilities, $P(X \mid Y)$, $P(Y)$, $P(X)$ when it is the probability $P(Y \mid X)$ that is required. This theorem is central to Bayesian statistics, which calculates the probability of a new event on the basis of earlier probability estimates derived from empirical data. The following section explains the different ways of performing statistical analyses using the classical and the Bayesian statistics.

4.5 Classical vs. Bayesian statistics

4.5.1 Using statistics

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

4.5.2 Using statistics

Consider the scores from one hundred students that have taken a test. Each test is marked with a score between zero and fifty. The collected data is somewhat uninformative as it is merely a list of numbers. To improve the presentation it is possible to add up the number of people who achieved the same mark. This is called the frequency for each mark. For example, 3 people scored 13 points, 10 scored 21 points etc. This information can now be represented as a histogram, where the mark is assigned to the x-axis and the number of students with that mark along the y-axis. This presentation is called the frequency distribution. Frequency distributions are important in statistical analysis as they provide an informative representation of the data. Statistical tests can be applied to frequency distributions to answer research questions, to confirm or reject certain hypothesis.

4.5.3 Objective and subjective probabilities

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

Now consider the scenario where a Casino employee, who is an expert on biased dice, is present and claims that there is a 98% certainty that a particular die is biased. However, this additional information does not benefit the test as such subjective degrees of belief are ignored in classical statistics.

As opposed to classical, Bayesian statistics takes a subjective degree of belief into account, the prior data. It allows us to use the information offered by the expert in our prediction as to whether or not the die is biased. In fact many experts could be consulted and asked for their opinions. With Bayesian statistics it is possible to take subjective probabilities together with the collected data to obtain the probability of the die being biased.

4.5.4 Inference differences

Another difference between classical and Bayesian statistics is how their inference is performed. In classical statistics an initial assumption or the hypothesis about the research question is first made. It is usually called a null hypothesis. A single or several alternative hypotheses can also be defined. Then the relevant evidence or data are collected. This evidence measures how different the observed results are from the expected if the null hypothesis was true. The measurement is given in terms of a calculated probability called the *p*-value. It is the probability of obtaining the observation found in the collected data, or other observations which are even more extreme.

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

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

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

4.5.5 Example of statistical spam classification

In order to implement either the classical or the Bayesian statistics to classify an e-mail massage as spam, some data must be available. This usually occurs as a collection of e-mail messages labeled as spam or non-spam and are referred as a corpus (training data). In addition, the information about the frequency distribution of the tokens in the corpus is available. This means that every token is accompanied by the number of times it has been seen in the corpus.

4.5.5.1 Classical statistics

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

4.5.5.2 Bayesian statistics

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

5 Naive Bayesian Spam Filtering

5.1 The model

A general Naive Bayesian spam filtering can be conceptualized into the model presented in Figure 1. It consists of four major modules, each responsible for four different processes: message tokenization, probability estimation, feature selection and Naive Bayesian classification.

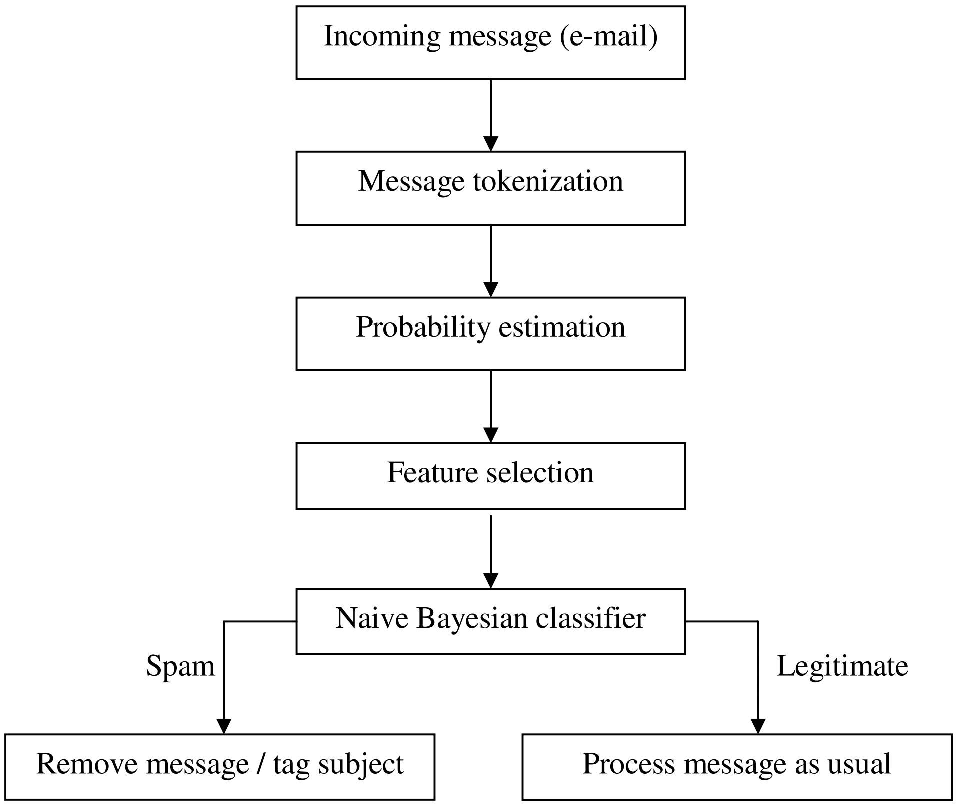

Figure 1. A model of Naive Bayesian spam filtering.

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

In the following text, the process of Naive Bayesian classification is described, followed by details concerning the measuring performance. This order of explanation is necessary because the sections concerned with the first three modules require understanding of the classification process and the parameters used to evaluate its improvement.

5.2 Naive Bayesian Classifier

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

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

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

This assumption is, however, not true, as words in e-mails are not independent. For example an e-mail with the word 'Viagra' is likely to co-occur with 'purchase'. However even though the independence assumption is not true the classifier works well, at least on the spam domain. One argument (Domingos & Pazzani 1996) is that with the independence assumption the classifier would produce poor probabilities, but the ratio between them would be approximately the same as would occur using conditional probabilities. Using the somewhat 'Naive' independence assumption gave birth to its name Naive Bayesian classifier.

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

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

Inserting (6) into (5) yields

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

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

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

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

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

The formal representation (9) may appear to be complicated. However, if $C = spam$ then (9) can basically be read as: “The probability of a message being spam given its features equals the probability of any message being spam multiplied by the probability of the features co-occurring in a spam divided by the probability of observing the features in any message”.

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

5.3 Measuring the performance

The meaning of a good classifier can vary depending on the domain in which it is used. For example, in spam classification it is very important not to classify legitimate messages as spam as it can lead to e.g. economic or emotional suffering for the user.

5.3.1 Precision and recall

A well employed metric for performance measurement in information retrieval is precision and recall. These measures have been diligently used in the context of spam classification (Sahami *et al.* 1998).

Recall is the proportion of relevant items that are retrieved, which in this case is the proportion of spam messages that are actually recognized. For example if 9 out of 10 spam messages are correctly identified as spam, the recall rate is 0.9.

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

Formally:

Let n_{gg} be the number of good messages classified as good (also known as false negatives).

Let n_{gs} be the number of good messages classified as spam (also known as false positives).

Let n_{ss} be the number of spam messages classified as spam (also known as true positives).

Let n_{sg} be the number of spam messages classified as good (also known as true negatives).

The precision (p) and recall (r) are defined as

$$p = \frac{n_{ss}}{n_{ss} + n_{gs}} = \frac{1}{1 + n_{gs} / n_{ss}} \tag{10}$$

$$r = \frac{n_{ss}}{n_{ss} + n_{sg}} = \frac{1}{1 + n_{sg} / n_{ss}} \tag{11}$$

The precision calculates the occurrence of false positives which are good messages classified as spam. When this happens p drops below 1. Such misclassification could be a disaster for the user whereas the only impact of a low recall rate is to receive spam messages in the inbox. Hence it is more important for the precision to be at a high level than the recall rate. The precision and recall reveal little unless used together. Commercial spam filters sometimes claim that they have an incredibly high precision value of 0.9999% without mentioning the related recall rate. This can appear to be very good to the untrained eye. A reasonably good spam classifier should have precision very close to 1 and a recall rate > 0.8.

A problem when evaluating classifiers is to find a good balance between the precision and recall rates. Therefore it is necessary to use a strategy to obtain a combined score. One way to achieve this is to use weighted accuracy.

5.3.2 Weighted accuracy

To reflect the difference in misclassifying a good message and a spam message a cost sensitive evaluation (Androutsopoulos *et al.* 2004) is used to measure the performance of the classifier. The weighted accuracy of a classifier is defined as

$$W_{Acc} = \frac{\lambda \cdot n_{gg} + n_{ss}}{\lambda \cdot n_g + n_s}. \tag{12}$$

where n_g is the total number of messages and n_s is the total number of spam messages. λ is the weight of each good message. Each misclassification of a good message counts as λ misclassifications of spam. At the same time, one correctly classified message counts as λ successful classifications. Spam messages are treated as single messages. Using weighted accuracy gives the impression that legitimate messages are λ times more important than spam.

5.3.3 Cross validation

There are several means of estimating how well the classifier works after training. The easiest and most straightforward means is by splitting the corpus into two parts and using one part for training and the other for testing. This is called the holdout method. The disadvantage is that the evaluation depends heavily on which samples end up in which set. Another method that reduces the variance of the holdout method is k -fold cross-validation.

In k -fold cross-validation (Kohavi 1995) the corpus, M , is split into k mutually exclusive parts, $M_1, M_2, \ldots M_k$. The inducer is trained on $M \setminus M_i$ and tested against M_i . This is repeated k times with different i such that $i \in \{1,2,\ldots,k\}$. Finally the performance is estimated as the mean of the total number of tests. For a k-folded test the precision p and the recall r are defined as

$$p = \frac{1}{n}\sum_{i=1}^{k} p_i \text{ and } r = \frac{1}{n}\sum_{i=1}^{k} r_i \qquad \text{(13 and 14)}$$

where p_i and r_i are the precision (10) and recall (11) for each of the k tests.

Research has shown that $k = 10$ is a satisfactory total (Breiman & Spector 1992) and (Kohavi 1995), therefore 10-fold cross validation was used throughout the experiments in this thesis.

5.3.4 Benchmark corpuses

One problem when attempting to benchmark a classifier arises because of the different results obtained depending on the test corpus. It is easy to optimize parameters for a classifier to favor a given corpus to achieve impressive results. When this classifier is applied to another corpus it may perform poorly. Therefore it is important to use more than one corpus source when testing a particular classifier. It is also preferred that corpuses are publicly available in case another researcher wants to compare or reproduce the results.

5.4 Message tokenization

Message tokenization is the process of tokenizing an e-mail message using carefully selected delimiters. The selection of delimiters affects the classification accuracy. Traditionally, selection is performed manually by trial and error. This thesis examines the possibility of automatic delimiter selection. The problem of finding high performance delimiters is called the feature subset problem. In this case the features refer to the delimiters.

There have been few attempts made to automatically find a good delimiter subset for spam classification. In fact, the author is only aware of one previous attempt (Bevilacqua-Linn 2003) which attempted to use two different search methods, a hill-climber and a genetic algorithm. Both search methods proved to be too time-consuming in accomplishing their task and the experiment was abandoned.

Since the dimensionality tends to be astronomically large, exhaustive search is not plausible. In this work Sequential Forward Floating Selection (SFFS) is implemented to find a good delimiter subset. There are two common schemes for implementing a search algorithm, called filters and wrapper induction and both approaches are examined and implemented in this thesis. For each method a fitness function is introduced. The fitness for the filter approach uses the discriminative power of the tokenized data. The wrapper fitness estimates the delimiter performance by running the inducer (measures the performance of the delimiters by running the classifier on test data).

This section begins by presenting some definitions after which it explains how delimiters interact. This explanation is necessary to decide upon the choice of search algorithm. Finally filters and wrappers are presented.

5.4.1 Definitions

The **alphabet** $A = \{a_1, a_2, ..., a_n\}$ is a finite set of symbols. Each symbol is a character. For example, the English alphabet $E = \{a, b, ..., z\}$ and the binary alphabet $B = \{0,1\}$. E-mail messages use ASCII character code. The alphabet used is $A_{ASCII} = \{char(0), char(1), ..., A, B, ..., (char255)\}$, where *chr(i)*, $0 \le i \le 255$ is the character corresponding to integer *i.*

A **string** X_A is a sequence of symbols from the alphabet A. This is denoted by $X_A = < s_1, s_2, ..., s_n >: s_i \in A$. For example, let the alphabet $A = \{a, m, p, s\}$ then a string over A could be $X_A = spam$. Every e-mail message is a string over the alphabet A_{ASCII}.

The **cardinality** of a string X is denoted by $|X| = n$. For example, if $X = spam$, then $|X| = 4$. The cardinality of the alphabet A_{ASCII} is $|A_{ASCII}| = 256$.

A **delimiter** set, D, for a string X over the alphabet A is a subset of A. For example, if the string *X=<Dear Friend how are you?>* then the delimiter set for X could as an example be *D={space, question-mark}.*

A **tokenizer** T is a function of a string X and a delimiter set D such that $Q = T(X, D)$. Where the **production** $Q = \{x_1, x_2, ..., x_n\} : x_i$ *is a substring between two delimiters in* X. For example, the string X=<*Dear Friend how are you?*> and the delimiter subset *D={space}*, then $T(X, D)$ *={Dear, Friend, how, are, you?}.*

A **label** set $L = \{l_1, ..., l_m\}$ is a finite set of labels, l_i, $1 \le i \le m$. The label set used for e-mail messages has only two labels, namely $L = \{good, spam\}$.

A **classifier** C is a function that maps a message X to a label l. This is defined as $C(X) = l \in L$.

Let $M_l = \{m_1, m_2, ..., m_n\} : C(m_i) = l$ be a **collection of previously labeled messages** which all have the same label.

5.4.2 Delimiter Interaction

An intuitive model of the delimiter interaction would be to think of it as a diagram where the tokenizer productions are plotted. Let the y-axis be the probability of spam (spaminess) of each token (x-axis). The spaminess of a token is calculated from its frequencies in the training data. (This is called the probability estimator and is examined in the next chapter.) If no delimiters are used, all tokens are likely to occur in the middle of the diagram, where they have a 50% probability of being spam or good related. As new delimiters are introduced the tokens will start to separate in the diagram, as if the tokens were attracted by magnets pulling them towards either the top (high spam probability) or bottom (low spam probability) of the diagram. The more separation in both directions often means a better delimiter subset.

Clearly, using no delimiters at all allows for a very high precision and a very low recall. As new delimiters are added the recall rate may increase to a particular level before the precision starts to fall. The aim is to choose delimiters that maximize the recall rate while maintaining a high precision. One reason for the precision fall is related to the fact that information is removed from the messages as the number of delimiters increases. This is called information loss. Information loss contributes to the destruction of patterns that characterize one class from another. For example, in unigram character tokenization every character is treated as a token. This separation of characters removes all patterns and the information loss is high. The result is low classifier performance.

The interaction between delimiters for a message tokenizer is highly complex; it is uncertain as to whether the relation between delimiters is transitive and also suffers from non-monotonicity. To demonstrate the non-transitivity and non-monotonicity properties of delimiter interaction the following simple example should be considered.

Let the alphabet $A = \{a, b, c, d\}$ and let the messages $X = abcd$ and $Y = bcad$.

Let the label of the messages be $L(X) = good$ and $L(Y) = spam$. Let the delimiters used by the tokenizer T be D.

Table 1 illustrates the productions $Q_1 = T(X, D)$ and $Q_2 = T(Y, D)$. Clearly a tokenizer that produces different productions depending on whether the message is good or spam is more beneficial than one that does not. This intuition can be used to demonstrate the non-transitive property. A simple criterion (fitness) function $J(D) = |Q_1 \cup Q_2| - |Q_1 \cap Q_2|$ is created by counting the number of different elements in Q_1 and Q_2.

Table 1. Illustration of the non-transitive relationship between delimiters.

D	Q_1	Q_2	$J(D)$
$\{a\}$	$\{bcd\}$	$\{bc, d\}$	3
$\{a, b\}$	$\{cd\}$	$\{c, d\}$	3
$\{b, c\}$	$\{a, d\}$	$\{ad\}$	3
$[a, c\}$	$\{b, d\}$	$\{b, d\}$	0

5.4.2.1 Non-transitivity

Let us denote a beneficial relationship between two features with $\rightarrow$. A transitive relation could appear as $a \rightarrow b \wedge b \rightarrow c \Rightarrow a \rightarrow c$ and is read as “the delimiter ‘a’ is of benefit to the classification together with ‘b’, ‘b’ is of benefit together with ’c’ therefore ‘a’ is of benefit with ‘c’”.

As shown the delimiters $\{a,b\}$ and $\{b,c\}$ are beneficial in the classification task since the productions of T are distinguishable. The presence of a transitive property would imply that the production of $\{a,c\}$ is also beneficial, but this is not the case as $a \rightarrow b \wedge b \rightarrow c \Rightarrow \neg a \rightarrow c$. This illustrates the fact that delimiter interaction is non-transitive.

5.4.2.2 Non-monotonicity

Monotonicity is the property that states that whenever a new feature is added the criterion function $J(D)$ shows an improvement. In mathematical notation the property is expressed as: Given two sets, D_1 and D_2, $D_1 \subset D_2 \Rightarrow J(D_1) < J(D_2)$. However, this is not the case for most criterion functions. If the previous example is considered, then $\{a\} \subset \{a,c\} \nRightarrow J(\{a\}) < J(\{a,c\})$. The non-monotonicity property for delimiter interaction has now been shown for the simple $J(D)$.

5.4.3 Dimensionality reduction in the search for a good delimiter subset

The order of delimiters is not of importance and neither are repetitions, hence the exhaustive search space for a delimiter subset of size s over an alphabet with cardinality n is $\binom{n}{s}$. The dimensionality for all subsets is 2^n. The growth is combinatorial and therefore an exhaustive search would be too slow for this purpose. Another choice is the branch and bound search which is much faster than an exhaustive search, but does require the monotonicity property. As shown previously in this chapter, finding a delimiter subset suffers from nonmonotonicity which makes the use of branch and bound algorithms inappropriate. As the delimiter interaction is non-transitive it is not possible to rely on transitive graphs to find maximum cliques. Other search space reduction algorithms are genetic algorithms (Siedlecki & Sklansky 1988).

Two other simple algorithms (Jain & Mao 1999), Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS), are very fast but are sub-optimal. SFS begins with no features and repeatedly adds the one that maximizes the fitness function. This algorithm only examines $n(n+1)/2$ subsets. SBS works in the same manner as SFS but it starts with all features and removes the feature that maximizes the fitness for the remainder.

In order to create results closer to the optimal solution (Pudil & Kittler 1994) improved SFS and SBS were required. Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS) are, respectively, the improved algorithms. SFFS works in a similar manner to SFS, but for every new subset it enters a backtracking loop that attempts to find a better subset than that of its predecessor by removing one feature at a time. This is repeated until no better subset is found and the backtrack loop is then exited. These algorithms gave near optimal results in some experiments (Pudil & Kittler 1994, Jain & Zongker 1997, Ferri & Pudil 1994). The original SFFS and SBFS had one defect; it could dismiss superior subsets when switching from the backtracking loop. An adaptive version of SFFS (Somol *et al.*1999) was proposed to address this problem. The solution was to keep a record of all the best subsets. After backtracking, instead of presuming that the forward subsets are better than the previous ones, a check is performed against the record. Figure 2 shows the pseudo code for the modified SFFS.

Due to the previous impressive results the adapted version of SFFS was implemented to find a good subset of delimiters. There is a small tradeoff in speed in comparison with SFS. The running time may increase by a factor of 2 to 10.

Modified SFFS
Input: A set of features, $F = \{f_1, f_2, ..., f_n\}$.
Input: A criterion function $J(F)$ that outputs higher values for better subsets.
Output: A feature subset $S \subseteq F$ that maximizes J.

```
$k \leftarrow 0$
$S_0 \leftarrow \emptyset$
Forward loop while (k < n)
    Find feature $f_i$ that makes $\arg\max_i (J(S_{k+1}^1 = S_k \cup f_i)) : f_i \notin S_k$
    if $S_{k+1} = \emptyset$ or $J(S_{k+1}^1) > J(S_{k+1})$
        $S_{k+1} \leftarrow S_{k+1}^1$
    $k \leftarrow k + 1$

    Backtrack loop
        Find feature $f_i$ that makes $\arg\max_i (J(S_{k-1}^1 = S_k \setminus f_i)) : f_i \in S_k$
        if $J(S_{k-1}^1) > J(S_{k-1})$
            $S_{k-1} \leftarrow S_{k-1}^1$
            $k \leftarrow k - 1$
        else
            Leave backtrack loop
    End backtrack loop
End forward loop
```

Figure 2. Pseudo code inspired by (Spance & Sajda 1998) for the modified SFFS.

5.4.4 Filters and wrappers

The two most common feature selection methods are filters and wrapper induction.

The filter approach (Kohavi & John 1996), does not use the inducer to estimate the fitness of features, it only analyses the data distribution. Hence filters are relieved from bias and errors in the inducer and may be able to find a more general solution. Filter methods generally run faster than wrappers. Two well-known filters are FOCUS (Almuallim & Dietterich 1991), and Relief (Kira & Rendell 1992). Figure 3 illustrates the filter procedure.

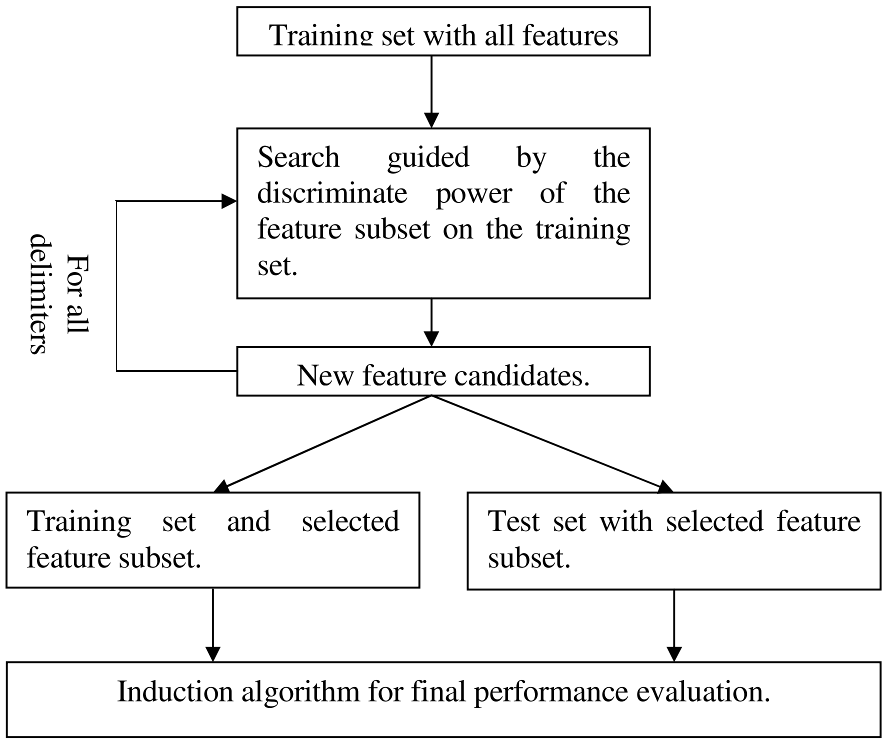
Training set with all features
Search guided by the discriminate power of the feature subset on the training set.
For all delimiters
New feature candidates.
Training set and selected feature subset.
Test set with selected feature subset.
Induction algorithm for final performance evaluation.

Figure 3. Illustration of the filter selection procedure.

Wrapper induction (Kohavi & John 1996, Kushmerick 1997 & 1999) uses the inducer as a part of the fitness function. Wrapper induction has successfully been used for information extraction of Internet resources such as the stock market and product catalogs. This is closely related to the purpose of this thesis. Their aim was to find delimiters in HTML code for better information retrieval, whereas this work attempts to find message delimiters for better spam classification. Figure 4 illustrates the wrapper model. Wrappers often produce better results than filters but they may be less general.

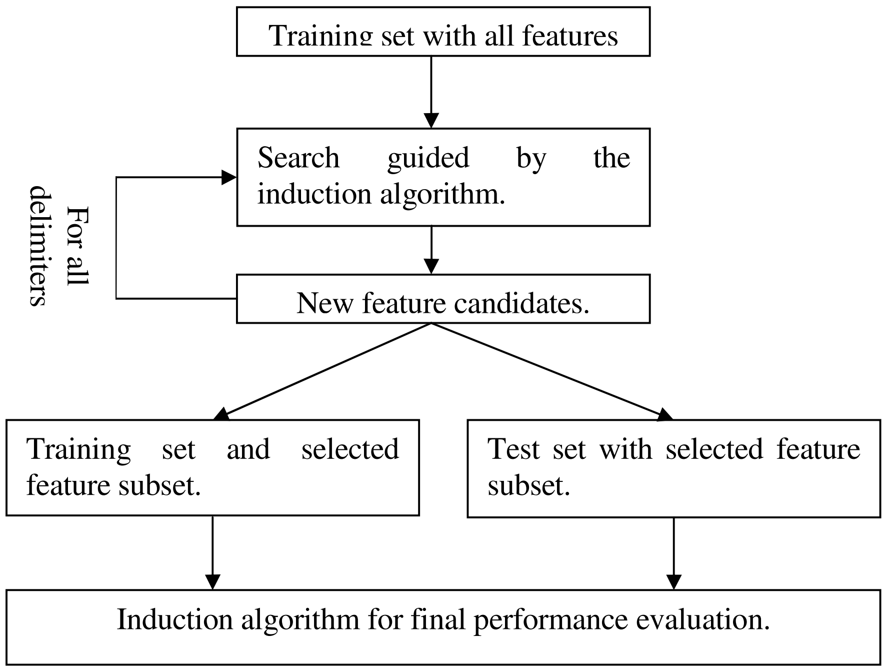
Training set with all features
Search guided by the induction algorithm.
For all delimiters
New feature candidates.
Training set and selected feature subset.
Test set with selected feature subset.
Induction algorithm for final performance evaluation.

Figure 4. Illustration of the wrapper selection procedure.

5.5 Probability estimation

The most straightforward approach for estimating the probability of an item is to divide its count by the total counts. This estimate approaches the theoretical probability as the sample space increases. If the entire population is available an exact probability can be obtained, but for most cases when the sample space is finite the Maximum Likelihood Estimate (MLE) will be in the neighborhood of the probability.

$$P_{MLE}(x_i) = \frac{|x_i|}{N} \tag{15}$$

where x_i is the item and N is the total number of training instances. The problem regarding this estimator is that it estimates the probability of unseen items to bee zero. This is sometimes referred to as the zero-frequency-problem. In the case of a Naive Bayesian classifier where estimates are combined from two corpuses a single zero estimate of a token could result in an overall spam probability of 0. The same problem remains for rare words, those with low frequencies. For example a probability estimated from a token occurring once in the good corpus and ten times in the spam corpus is less reliable than a token with 10 occurrences in the good and 100 in the spam corpus. Even though the combined probability is the same the latter is more reliable.

Schemes do exist to solve the problems concerning Maximum Likelihood Estimation for cases with sparse data. It is usually said that these schemes perform smoothing. Smoothing is the method of moving probability mass from seen to unseen items.

The labels used are given below.

N is the total number of training instances.

x_i is the item.

N_j is the frequency of frequency of items seen j times. For example, if there exist exactly three tokens which each have been seen five times then $N_5 = 3$.

B is the number of bins (distinct items).

5.5.1 Absolute Estimate (p_{abs})

The absolute estimate (Nay *et al.* 1994), subtracts a constant from the frequencies

$$P_{Abs}(x_i) = \frac{|x_i| - c}{N} \tag{16}$$

where c is the constant usually obtained from the frequency of frequency of items seen once and twice, N_1 and N_2 such that

$$c = \frac{N_1}{N_1 + 2N_2} \tag{17}$$

5.5.2 Laplace Estimate (p_{lap})

Laplace (or add-one) is perhaps the simplest smoothing technique which assumes that every event has been seen once before.

$$P_{Lap}(x_i) = \frac{|x_i| + 1}{N + B} \tag{18}$$

Where B is the number of bins (distinct items). The problem with the Laplace estimator is that it allocates too much probability mass to unseen objects (Gale & Church 1994).

5.5.3 Expected Likelihood Estimate (ELE) (p_{ELE})

This method assigns less probability mass to unseen objects than the Laplace estimator by adding 0.5 instead of 1 to every count.

$$P_{ELE}(x_i) = \frac{|x_i| + 0.5}{N + 0.5B} \tag{19}$$

This estimator is also referred to as Jeffreys-Perks law.

5.5.4 Lidstone Estimate (p_{Lid})

Laplace and ELE are both special cases of the Lidstone estimator which adds δ to every count

$$P_{Lid}(x_i) = \frac{|x_i| + \delta}{N + \delta B} \tag{20}$$

where, usually, $0 \le \delta \le 1$. This estimate works in a similar manner as the Laplace estimate and ELE. The difference is that instead of using a constant to move the probability mass to unseen items, it uses an adjustable variable, δ. But its weakness lies in finding a good δ.

5.5.5 Witten Bell smoothing (p_{WB})

Witten Bell smoothing (Witten & Bell 1991) was originally developed for the task of text compression. It uses items seen once to estimate unseen items. The probability mass assigned to unseen items is

$$P(x_i) = \frac{N_1}{N_1 + N} \text{ for } |x_i| = 0 \tag{21}$$

where N_1 is the number of items that have been seen once. The mass is taken from higher frequency items such that

$$p_{wb}(x_i) = \frac{|x_i|}{N + N_1} \text{ for } |x_i| > 0 \tag{22}$$

The probability for unseen items is determined by

$$p_{wb}(x_i) = \frac{N_1}{Z(N + N_1)} \text{ for } |x_i| = 0 \tag{23}$$

where Z is the number of unseen items which can be derived by $Z = B - N_1$.

5.5.6 Good Turing Estimate (p_{SGT})

Good Turing estimators (Good 1953) use the following equation to calculate the probability of events that have been observed before

$$P_{GT}(x_i) = \frac{r^*}{N} \text{ where} \tag{24}$$

$$r^* = (r+1) \cdot \frac{E(N_{r+1})}{E(N_r)} \tag{25}$$

where $E(n)$ is the expectation of the variable n. It estimates how many different words were seen n times, r is the frequency of the item, N_r is the frequency of that frequency and r^* is the new estimated frequency. There are many different Good Turing estimators depending on how the E function is selected. For the experiment a version[16] called Simple Good-Turing (Gale 1995), denoted as p_{SGT} was used.

5.5.7 Bayesian smoothing (p_{Rob})

Finally an approach that smoothes (Robinson 2003) the maximum likelihood estimate by assuming a beta distribution for the prior was examined. This estimate was first adapted to smooth spam probabilities by Gary Robinson. Since then it has been implemented in a number of spam filters[17] and has shown very promising results.

[16] Thanks to Geoffrey Sampson at the University of Sussex who has made the source code for the Simple Good Turing estimator freely available at his homepage, http://www.grs.u-net.com/.

[17] SpamBayes and BogoFilter both uses Robinsons approach.

$$P_{Rob}(x_i) = \frac{s \cdot p_0 + n \cdot p_{MLE}(x_i)}{s+n} \tag{26}$$

where p_0 is the expected probability when there is no data, n is the number of data points, s is the strength of the smoother.

Diagram 1 illustrates how this technique smoothes $p_{MLE}(x_i) = 0.8$ for two different strengths $s = 1.0$ and $s = 0.5$ as the number of data points increase. As seen in the table, when there are no data points a probability of 0.5 is assigned to the data ($p_0 = 0.5$).

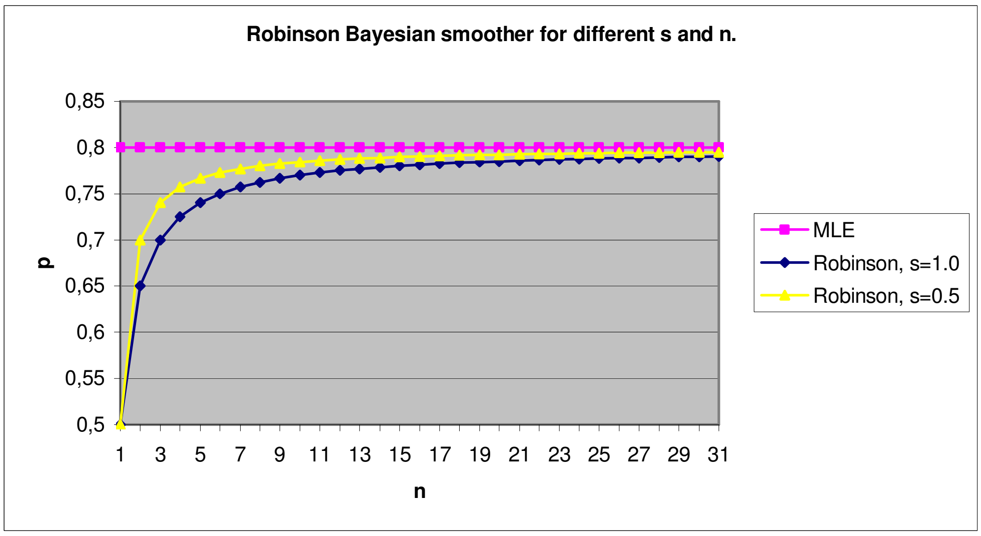

Diagram 1. $p_{MLE}(x_i) = 0.8$ is smoothed by p_{Rob} as the data points increase.

5.6 Feature Selection

The primary purpose of feature selection is to reduce the dimensionality to decrease the computation time. This is particularly important concerning text categorization where the high dimensionality of the feature space is a problem. In many cases the number of features is in the tens of thousands. Then it is highly desirable to reduce this number, preferably without any loss in accuracy. Several feature selection methods have been proposed. The χ^2 statistics, Information Gain, Mutual Information, Term Strength, Document Frequency, Probability Ratio and Odds Ratio are some of them. A number of studies have been conducted to compare their performance. A comparative study (Yang & Pedersen 1997) showed that Information Gain, χ^2 and Document Frequency had excellent performance on Reuters corpus using a k-Nearest Neighbor classifier. Another study (Mladenic & Grobelnik 1999) showed that Odds-Ratio outperformed Information Gain using a Naive Bayesian classifier due to the different domains and classifier. (Forman 2003) strengthens the results of (Yang & Pedersen 1997) and introduced a new feature selection method, BNS which showed promising results. Another purpose of feature selection is to improve performance when using all features. This is rarely the case, but it is possible. For example, (Forman 2003) showed that bi-normal separation (BNS) did improve the results when using all features.

However, when dealing with spam, it is often a matter of several hundred or perhaps thousands of features as opposed to hundreds of thousands in other areas of text categorization. In comparison to other classifiers such as neural networks, the computations in a Naive Bayesian classifier are rapid. This work will also investigate whether there is a need for feature selection when a Naive Bayesian spam classifier is used for spam detection.

Three common feature selection methods will be investigated, Information Gain, χ^2 and the Probability Ratio. Odds Ratio and Probability Ratio are closely related. The reason for using these three feature selection methods in this study is that they are commonly used in spam classification and have all shown good performance. A brief description about each feature selection method is given in the following subsections.

Information Gain

Information gain measures the decrease in entropy, or the number of bits gained knowing a feature is present or absent. The information gain, IG , for a term t is defined (Yang & Pedersen 1997) as

$$IG(t) = \sum_{X \in \{t,\bar{t}\}} \sum_{Y \in \{c_i\}} P(X,Y) \log\left(\frac{P(X,Y)}{P(X) \cdot P(Y)} \right) \tag{27}$$

where $\{c_i\}_{i=1}^{m}$ denote the set of classes, in our case, good and spam.

5.6.1 χ^2 statistics

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

The chi square statistics is defined as

$$\chi^2 = \sum_{i,j} \left(\frac{(O_{ij} - E_{ij})^2}{E_{ij}} \right), \tag{28}$$

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

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

We can calculate the χ^2 value from

$$\chi^2 = \frac{N \cdot (AD - CB)^2}{(A+C) \cdot (B+D) \cdot (A+B) \cdot (C+D)}. \tag{29}$$

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

$$\chi^2_{\max} = \arg\max_i^m \{\chi^2(t, c_i)\} \tag{30}$$

5.6.2 Probability Ratio

Probability ratio encourages extreme probabilities and is highly dependent on the estimator. The feature weight is calculated by

$$PR(t) = \left|\log\left(\frac{p(t \mid good)}{p(t \mid spam)}\right)\right|. \tag{31}$$

In an anti-spam context the Probability Ratio has previously been used in (Graham 2002), but the probability ratio was measured as

$$PR_{Gra}(t) = \left|p(spam \mid t) - 0.5\right|. \tag{32}$$

Since Graham 2002 was published, Probability Ratio has become very popular in the implementation of spam filters due to its promising performance and simplicity. This work examines how well Probability Ratio compares to the more sophisticated Information Gain and χ^2 feature selection methods.

6 Experimental Results

C++ provided most of the necessary tools to carry out the experiments. The object oriented nature together with a dynamically loaded XML file allowed a setup of complex test configurations. All parameters and algorithms were specified in the XML. No recompilations were required between different tests. The only C++ shortcoming was its lack of high precision data types. In many cases the native double (64-bit) caused underflow which means that the value is rounded down to zero. When a spam token probability becomes zero, (9) will also become zero since the probabilities are multiplied. The solution was to deal with the mantissa and exponent separately.

Throughout this work seven corpuses are used for testing, the authors personal corpus, SpamAssassin[18], Ling-Spam (Androutsopoulos *et al.* 2000b), PU1, PU2, PU3 and PUA[19] (Androutsopoulos *et al*. 2004).

The personal corpus contains 930 good and 876 spam messages received by the author. All messages are intact i.e. in their original state. SpamAssassins public available corpus contains 3900 good and 1897 spam messages. They are published in their complete form. The Ling-Spam corpus consists of 2412 good messages received by its author and 481 spam messages from the public archives of the Linguists list. Finding benchmark corpuses with good messages has been problematic due to privacy aspects. The PU corpuses are an attempt to bypass the privacy issues of publishing legitimate e-mails. Each token is mapped to a number, encoding all messages in this manner creates a corpus involving no exploitation of privacy but which still maintains the original frequency distribution. Since the PU corpuses have been tokenized, they are not applicable for tokenization experiments. However, one advantage of being pre-tokenized is that they are a good platform for classifier performance benchmarking.

Table 2 presents the corpuses, their acronym, the number of good (n_g) and spam messages (n_s), the ratio and a description of the parts included in the corpus.

Table 2. Corpuses used in the experiments.

Corpus	Acronym	n_g	n_s	$n_g : n_s$	Description
Personal	P	930	876	1.06	Full messages
SpamAssassin	SA	3900	1897	2.05	Full messages
Ling-Spam	L-S	2412	481	5.01	Body+subject
PU1	PU1	618	481	1.28	Full messages
PU2	PU2	579	142	4.08	Full messages
PU3	PU3	2313	1826	1.27	Full messages
PUA	PUA	571	571	1.00	Full messages

[18] Spamassasins public corpus is available at http://spamassassin.apache.org/publiccorpus/.

[19] Ling-Spam, PU1, PU2, PU3, PUA public corpuses are available at http://www.aueb.gr/users/ion/ and is provided by Ion Androutsopoulos.

This chapter is divided into three sections, one for each experiment. The experiments are independent and can be read separately. The results in each section are followed by an analysis, conclusion and some possibilities concerning future work.

6.1 Delimiter selection

The purpose of this experiment was to find an improved delimiter subset via filter and wrapper guided searches. To accomplish this it was necessary to build a fitness function for both approaches.

6.1.1 Filter for delimiter selection using KL-divergence

A simple filter was created as a function of the probability distribution of the tokenized data. The output of the filter is a value representing the fitness of a delimiter set D. The criterion (or fitness) function is denoted by $J(D)$ and the search algorithm attempts to maximize it for the delimiter set.

Let
$M_g = \{g_1, g_2, ..., g_{n_g}\} : L(g_i) = good$ and $M_s = \{s_1, s_2, ..., s_{n_s}\} : L(s_i) = spam$ be the training data.

Let $T(m_i, D) = Q$ be the tokenizer. The two productions $T(M_g, D) = Q_1 = \{x_1, x_2, ..., x_n\}$ and $T(M_s, D) = Q_2 = \{y_1, y_2, ..., y_n\}$ contain the tokenized data (tokens) for M_g and M_s.

The probability estimator assigns a probability indicating the token spaminess for each token in Q_1 and Q_2 from their frequency distribution. The distance between the two respective discrete probability distributions, $q_1(x)$ and $q_2(x)$, is measured and used as the criterion function for D.

This filter implements the Kullback-Liebler divergence (or relative entropy) to measure the distance between the two probability mass functions, $q_1(x)$ and $q_2(x)$. The KL-divergence $KL(q_1 \| q_2)$ is defined as

$$KL(q_1 \| q_2) = \sum_x q_1(x) \log \frac{q_1(x)}{q_2(x)}. \tag{33}$$

$KL(q_1 \| q_2)$ is always non-negative and is zero if and only if $q_1 = q_2$. Furthermore, it is not symmetric but can be made so by calculating the two-way divergence

$$KL_s(q_1 \| q_2) = KL(q_1 \| q_2) + KL(q_2 \| q_1). \tag{34}$$

The greater the distance between q_1 and q_2 the better the delimiter set D; hence, the criterion function is set as

$$J_{KL}(D) = KL_s(q_1 \| q_2). \tag{35}$$

Even though it is independent of the inducer, the drawback of this filter is that it depends upon the probability estimator.

6.1.2 Wrapper approach for delimiter selection

The wrapper uses the inducer as a part of the criterion function. The inducer in this work uses stratified 10-fold cross-validation to establish more reliable results.

Let $M_g = \{g_1, g_2, ..., g_{n_g}\} : L(g_i) = good$ and $M_s = \{s_1, s_2, ..., s_{n_s}\} : L(s_i) = spam$ be the training data. Let $I(M_g, M_s, D)$ be the inducer as a function of the training data and a delimiter set D.

The criterion function $J(D)$ returns a weight which represents how well the delimiter set D contributes to the classification. For the experiments an extended version of Weighted Accuracy, W_{Acc}, defined in (12) for $\lambda = 1$ was used.

This heuristic could guide the search for a delimiter set to a certain level. Over time, W_{Acc} will have the same value. This happens when future subsets do not affect the precision or recall. The problem arises when one or more $J(D_{i+1})$ exists such that $J(D_i) = J(D_{i+1})$, where D_i is a delimiter subset of size i. The solution is to increase the granularity of W_{Acc} by introducing several new parameters in order to obtain the Extended Weighted Accuracy, where the mean values of the already classified data will be used. This is performed in the following way.

Let μ_g be the mean spam probability for all M_g and μ_s be the mean probability of spam for all M_s such that

$$\mu_g = \frac{1}{n_g} \sum_i^{n_g} P_s(g_i) \tag{36}$$

$$\mu_s = \frac{1}{n_s} \sum_i^{n_s} P_s(s_s) \tag{37}$$

The two means are combined with the F-measure (Rijsbergen 1979) which is the harmonic mean such that

$$\mu_{gs} = \frac{2 \cdot (1-\mu_g) \cdot \mu_s}{(1-\mu_g) + \mu_s} \tag{38}$$

Finally μ_{gs} is incorporated into W_{Acc} to obtain the Extended Weighted Accuracy:

$$EW_{Acc} = \frac{\lambda \cdot n_{gg} + n_{ss}}{\lambda \cdot n_g + n_s} + \alpha \cdot \mu_{gs} \tag{39}$$

where α is a small value to make μ_{gs} a faction of the Extended Weighted Accuracy. For this work the influence of μ_{gs} is set to be less important than W_{Acc}, that is $\alpha \cdot \mu_{gs} < \frac{1}{\lambda \cdot n_g + n_s}$. Since $0 \le \mu_{gs} \le 1$ it is possible to set $\alpha = \frac{1}{10 \cdot (\lambda \cdot n_g + n_s)}$. This means that EW_{Acc} is foremost guided by W_{Acc}. When W_{Acc} no longer changes, EW_{Acc} will instead be guided by μ_{gs}.

The Extended Weighted Accuracy function was implemented as the criterion function for the wrapper.

$$J_{EW_{Acc}}(D) = EW_{Acc}. \tag{40}$$

6.1.3 Experimental settings

The tests were carried out on three corpuses, Ling-Spam, SpamAssassin and the author's personal corpus. Since PU1, PU2, PU3 and PUA are pre-tokenized it was not possible to use them for this experiment. The size of the SpamAssassin corpus was cut down to 25% since the wrapper algorithm would otherwise have taken too long.

For simplicity, only single characters are used as delimiters, even though ordered clusters of characters could be beneficial. Further low frequency delimiters are also pruned since their affect on the outcome is minimal. It was found that delimiters that occurred less than five times in each message could be removed without affecting the performance significantly. A further restriction was to only use non alphanumerical characters as delimiter candidates.

Both the body and header part of messages are tokenized. HTML and plain message bodies are tokenized in the same manner. Attachments are removed prior to the tokenization process. Case is preserved and no word stemming is applied.

All tests were conducted with stratified 10-fold cross-validation. If there are several maximal solutions Occams Razor[20] is applied to choose the simplest. The simplest delimiter set is defined as the one with lowest cardinality.

Robinsons Bayesian smoother described in 5.5.7 was used since it works both rapidly and satisfactorily. Two base-lines were established, the first only using space as the delimiter and the second using all non alphanumerical characters as delimiters, as presented in Table 3. As it was

[20] Occams Razor states that one should not make more assumptions than the minimum needed. It admonishes us to choose from a set of otherwise equivalent models of a given phenomenon the simplest one.

anticipated that a better subset would be found somewhere between a space and all non alphanumerical characters, it was natural to choose these two as base-line delimiters. Also, it appears to be customary (Graham 2003) to consider all non alphanumerical characters as delimiters.

Table 3. Delimiter subsets used as base-line.

Base-Line	**Description**
D_S	Only space is used as delimiter, char(32).
D_{NAN}	All non alpha numerical characters are used as delimiters.

6.1.4 Results

Table 4 displays the delimiter subsets found by SFFS for the filter (J_{KL}) and wrapper $(J_{EW_{Acc}})$ algorithms. The subsets are denoted by D_{KL} and $D_{W_{Acc}}$ respectively. The delimiters are separated by space. The delimiter space is labeled as 'space'.

Table 4. Delimiters subsets automatically found by the SFFS for different corpuses.

Criterion	D_J	$D_J = SFFS(D,J) : D_J \subseteq D$
		Ling-Spam
J_{KL}	D_{KL}	space - / _ * =
$J_{EW_{Acc}}$	$D_{W_{Acc}}$	space) = _ (*
		SpamAssassin
J_{KL}	D_{KL}	space . = / - @ % _ < , + ; '
$J_{EW_{Acc}}$	$D_{W_{Acc}}$	space - ‘
		Personal
J_{KL}	D_{KL}	space / \ r + . - = _ @ : , (;)
$J_{EW_{Acc}}$	$D_{W_{Acc}}$	space . ,

Table 5 to 7 display the performances of the delimiters from Table 4 on the three corpuses. They also shows the performance of the base-line delimiters, D_S and D_{NAN}. The performance is shown as precision (p), recall (r) and Weighted Accuracy $(WAcc)$ for $\lambda = 1$, $\lambda = 9$ and $\lambda = 99$.

Table 5. Performance of the different delimiter subsets on Ling-Spam.

Ling-Spam	$\lambda = 1$			$\lambda = 9$			$\lambda = 99$		
Delimiter subset	p	r	W_{Acc}	p	r	W_{Acc}	p	r	W_{Acc}
D_S	91.71	98.96	98.34	92.61	98.96	98.44	93.87	98.75	98.71
D_{NAN}	93.32	98.75	98.62	94.43	98.75	98.84	95.19	98.75	99.00
D_{KL}	93.32	98.75	98.62	94.43	98.75	98.84	95.38	98.75	99.05
$D_{EW_{Acc}}$	94.26	98.96	99.82	95.20	98.96	99.00	96.15	98.75	99.21

Table 6. Performance of the different delimiter subsets on SpamAssassin.

SpamAssassin	$\lambda = 1$			$\lambda = 9$			$\lambda = 99$		
Delimiter subset	p	r	W_{Acc}	p	r	W_{Acc}	p	r	W_{Acc}
D_S	100.00	92.83	97.65	100.00	92.41	99.61	100.00	91.98	99.96
D_{NAN}	99.52	86.92	95.58	99.52	86.71	99.12	99.51	86.29	99.73
D_{KL}	99.30	90.08	96.55	99.30	90.08	99.20	99.30	89.24	99.64
$D_{EW_{Acc}}$	100.00	94.09	98.07	100.00	93.88	99.69	100.00	93.04	99.97

Table 7. Performance of the different delimiter subsets on Personal.

Personal	$\lambda = 1$			$\lambda = 9$			$\lambda = 99$		
Delimiter subset	p	r	W_{Acc}	p	r	W_{Acc}	p	r	W_{Acc}
D_S	98.02	99.00	98.50	98.02	99.00	98.10	98.02	99.00	98.01
D_{NAN}	98.81	83.00	91.00	98.80	82.00	97.30	98.77	80.00	98.81
D_{KL}	98.84	85.00	92.00	98.82	84.00	97.50	98.81	83.00	98.84
$D_{EW_{Acc}}$	100.00	99.00	99.50	100.00	99.00	99.90	100.00	99.00	99.50

6.1.5 Analysis

Using only space (D_S) proved to be more effective than using all non-alphanumerical (D_{NAN}) as delimiters except on the Ling-Spam corpus ($\lambda = 9$). This may be due to the different characteristics of the corpuses; the Ling-Spam corpus is stripped from its header, while the messages in the Personal and SpamAssassin are contained in their entirety. The decomposition of headers when using D_{NAN} may result in lower performance which would explain the differences.

The wrapper approach did find subsets of delimiters that significantly improved the overall classification performance on all corpuses. The filter approach using the KL divergence did not improve the performance on the subsets.

Both approaches (particularly the wrapper) found delimiter subsets with low cardinality for all corpuses. For, $D_{EW_{Acc}}$ for both the Personal and SpamAssassins' corpus consisted only of three delimiters (see Table 4).

Even though D_{NAN} worked slightly better than D_S on Ling-Spam, an even better low cardinality subset, $D_{W_{Acc}}$, was found by the wrapper. This suggests that a near-optimal solution is likely to consist of only a few delimiters. The reason for this is probably due to the information loss as the number of delimiters increases (see section 5.4.2). This is illustrated in Diagram 2 where EW_{Acc} is plotted against k, where s_k is the optimal delimiter subset of size k. Diagram 2 shows how the performance decreases as the number of delimiters increases for EW_{Acc}. The size of the delimiter subset is k.

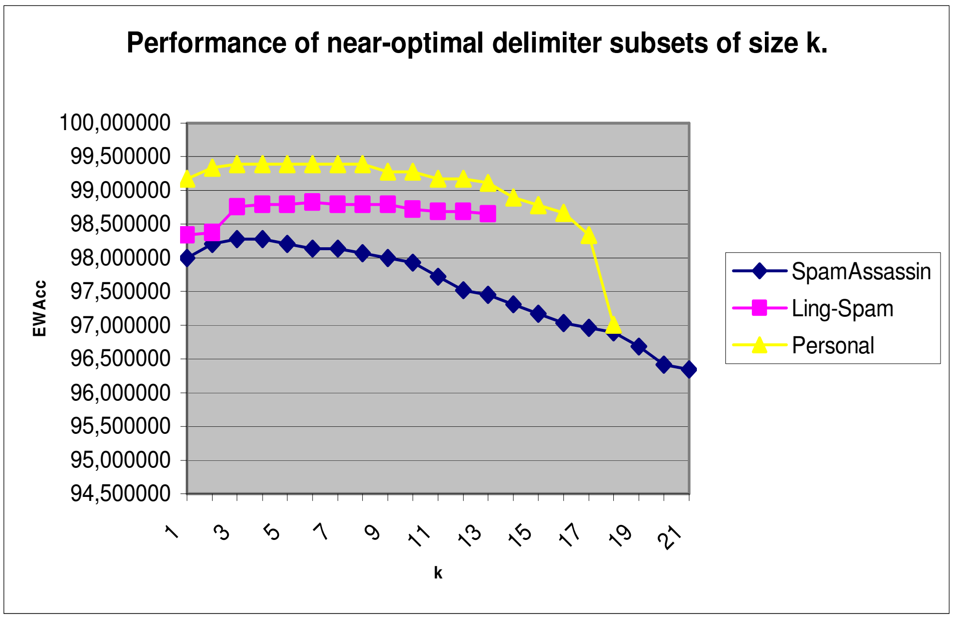

Diagram 2. Performance as the number of delimiters increase.

The subsets $D_{EW_{Acc}}$ had little in common apart from the fact that space was always found to be the optimal subset for $k = 1$.

On a 1.7GHz Intel Pentium machine it took about 13 hours to run the wrapper on 25% of the SpamAssassin corpus, the filter ran about ten times faster. The code is not optimized and can probably be made to run twice as fast, by using tokenization caching and other techniques.

6.1.6 Conclusion

In this chapter a near-optimal search algorithm called SFFS was applied to find a subset of delimiters for the tokenizer. Then a filter and a wrapper algorithm were proposed to determine how beneficial a group of delimiters is to the classification task. The filter approach ran about ten times faster than the wrapper, but did not produce significantly better subsets than the base-lines. The wrapper did improve the performance on all corpuses by finding small subsets of delimiters. This suggested an idea concerning how to select delimiters for a near-optimal solution, namely to start with space and then add a few more. Since the wrapper generated subsets had nothing in common apart from space, the recommendation is to only use space as a delimiter. The wrapper was far too slow to use in spam filter software.

6.1.7 Future work

In this work it was found that near-optimal solutions are likely to have low cardinality. Even though the wrapper was too slow to use at run time in spam filter software, it is believed that a near-optimal solution can be found by using SFS for small delimiter subsets. Such an algorithm could be included with the spam filtering software and executed by the user to tune the delimiter subset. During testing an attempt was made to plug $\lambda = 99$ into $J_{EW_{Acc}}$ which resulted in a completely different delimiter subsets (without space) yielding a precision of 100% and recall of approximately 60% on the test corpuses. It would be interesting to implement this under more controlled experiments. Another idea is to use different delimiters on the header and body of messages, due to their different characteristics. Also, in natural language it is common to use n-grams instead of unigrams for ambiguity resolution. Even though recent studies (Androutsopoulos 2004) found no evidence of any benefit from using n-grams for spam classification it was pointed out that further research is necessary.

6.2 Probability estimation

In this section the probability estimators described in 5.5 were tested against all corpuses.

6.2.1 Experimental settings

The performance was measured in the weighted accuracy using $\lambda = 1$, $\lambda = 9$ and $\lambda = 99$. All tests were conducted with stratified 10-fold cross-validation. In order to avoid biased feature selection the classifier was fed with all features. The same set of delimiters was used throughout all the tests. For the Lidstone estimate (p_{Lid}) a value of $\delta = 0.1$ was found to work well for the given corpuses. For the Bayesian Smoother (p_{Rob}) the strength of $s = 0.5$ and expected probability for unseen items $p_0 = 0.5$ was used. These values were found by trial and error. Robinson describes (Robinson 2003) how these parameters can be set automatically, unfortunately there were time constraints upon the author and this was not possible.

6.2.2 Results

The results are displayed in Table 8 to Table 13. Each table displays the performance of an estimator on a corpus. The performance is shown as precision (p), recall (r) and Weighted Accuracy $(WAcc)$. Each estimator is also ranked based on $WAcc$ to simplify the interpretation of the tables.

Table 8. Probability estimators tested on PU1.

PU1	$\lambda = 1$				$\lambda = 9$				$\lambda = 99$			
Est.	p	r	$WAcc$	R	p	r	$WAcc$	R	p	r	$WAcc$	R
p_{abs}	94,41	98,34	96,72	6	94,78	98,13	95,98	6	95,72	97,71	96,61	6
p_{lap}	93,12	98,54	96,18	7	93,86	98,54	95,27	7	94,4	98,13	95,49	7
p_{ELE}	94,78	98,13	96,82	5	95,74	98,13	96,72	5	95,92	97,71	96,77	5
p_{Lid}	96,32	97,92	97,45	4	97,11	97,71	97,73	4	97,30	97,51	97,89	3
p_{WB}	97,32	98,13	98,00	1	97,92	97,92	98,35	1	97,91	97,51	98,38	2
p_{SGT}	97,13	98,34	98,00	1	97,32	98,13	97,91	2	98,11	97,09	98,53	1
p_{Rob}	96,53	98,34	97,73	3	97,11	97,92	97,75	3	97,11	97,71	97,73	4

Table 9. Probability estimators tested on PU2.

PU2	$\lambda=1$				$\lambda=9$				$\lambda=99$			
Est.	p	r	$WAcc$	R	p	r	$WAcc$	R	p	r	$WAcc$	R
p_{abs}	88,36	90,85	95,84	6	89,44	89,44	97,2	7	90,58	88,03	97,73	7
p_{lap}	90,71	89,44	96,12	5	92,65	88,73	98,02	6	94,57	85,92	98,76	5
p_{ELE}	95,42	88,03	96,81	1	96,03	85,21	98,77	3	96,77	84,51	99,27	3
p_{Lid}	97,39	78,87	95,42	7	97,30	76,06	98,86	2	97,25	74,65	99,42	2
p_{WB}	92,75	90,14	96,67	3	93,38	89,44	98,21	5	94,07	89,44	98,60	6
p_{SGT}	94,07	89,44	96,81	1	94,74	88,73	98,52	4	94,70	88,03	98,76	4
p_{Rob}	98,35	83,8	96,53	4	98,33	83,10	99,22	1	98,31	81,69	99,61	1

Table 10. Probability estimators tested on PU3.

PU3	$\lambda=1$				$\lambda=9$				$\lambda=99$			
Est.	p	r	$WAcc$	R	p	r	$WAcc$	R	p	r	$WAcc$	R
p_{abs}	97,95	96,93	97,75	5	98,05	96,55	98,33	5	98,21	96,22	98,60	5
p_{lap}	96,54	97,86	97,51	7	96,85	97,54	97,50	7	97,10	97,32	97,71	7
p_{ELE}	97,43	97,65	97,83	3	97,69	97,26	98,11	6	97,84	96,88	98,30	6
p_{Lid}	98,12	97,26	97,97	2	98,28	96,93	98,52	3	98,38	96,50	98,73	3
p_{WB}	98,71	96,28	97,80	4	98,70	95,67	98,74	1	98,86	95,29	99,10	1
p_{SGT}	98,60	96,17	97,70	6	98,65	95,73	98,70	2	98,81	95,18	99,06	2
p_{Rob}	98,12	97,37	98,02	1	98,17	96,99	98,45	4	98,32	96,39	98,68	4

Table 11. Probability estimators tested on PUA.

PUA	$\lambda=1$				$\lambda=9$				$\lambda=99$			
Est.	p	r	$WAcc$	R	p	r	$WAcc$	R	p	r	$WAcc$	R
p_{abs}	98,70	93,33	96,05	6	98,69	92,81	98,18	6	98,68	91,93	98,70	7
p_{lap}	98,54	94,56	96,58	1	98,88	93,33	98,39	2	98,87	92,11	98,88	1
p_{ELE}	98,71	94,21	96,49	2	98,89	93,86	98,44	1	98,87	92,11	98,88	1
p_{Lid}	98,89	93,68	96,32	3	98,88	93,16	98,37	3	98,87	92,46	98,88	1
p_{WB}	98,70	93,33	96,05	6	98,69	92,81	98,18	6	98,69	92,28	98,71	6
p_{SGT}	98,89	93,51	96,23	4	98,88	92,98	98,35	4	98,87	92,28	98,88	1
p_{Rob}	98,89	93,51	96,23	4	98,88	92,81	98,33	5	98,86	91,58	98,87	5

Table 12. Probability estimators tested on Ling-Spam.

L-S	$\lambda = 1$				$\lambda = 9$				$\lambda = 99$			
Est.	p	r	$WAcc$	R	p	r	$WAcc$	R	p	r	$WAcc$	R
p_{abs}	75,00	100,00	94,46	6	76,15	99,79	93,91	6	77,35	99,58	94,20	6
p_{lap}	73,31	99,58	93,91	7	74,11	99,58	93,21	7	76,11	99,58	93,79	7
p_{ELE}	85,03	99,38	96,99	5	86,23	99,17	96,90	5	87,18	99,17	97,10	5
p_{Lid}	93,14	98,96	98,62	1	94,43	98,96	98,84	1	94,99	98,75	98,96	1
p_{WB}	91,94	99,79	98,51	2	92,46	99,58	98,41	4	94,07	99,17	98,76	3
p_{SGT}	92,08	99,38	98,48	3	92,97	99,17	98,52	2	94,26	99,17	98,80	2
p_{Rob}	92,07	99,17	98,44	4	92,61	99,17	98,44	3	93,52	99,17	98,63	4

Table 13. Probability estimators tested on SpamAssassin.

SA	$\lambda = 1$				$\lambda = 9$				$\lambda = 99$			
Est.	p	r	$WAcc$	R	p	r	$WAcc$	R	p	r	$WAcc$	R
p_{abs}	99,20	91,88	97,10	4	99,20	91,72	99,24	5	99,20	91,51	99,60	7
p_{lap}	99,47	89,09	96,27	7	99,47	88,93	99,21	7	99,47	88,67	99,71	4
p_{ELE}	99,42	89,98	96,55	6	99,42	89,77	99,23	6	99,42	89,67	99,69	6
p_{Lid}	99,48	90,62	96,77	5	99,48	90,46	99,29	4	99,48	90,30	99,72	3
p_{WB}	99,50	94,94	98,19	1	99,50	94,94	99,52	1	99,50	94,89	99,75	1
p_{SGT}	99,34	94,94	98,14	2	99,39	94,89	99,47	2	99,39	94,83	99,69	5
p_{Rob}	99,49	92,30	97,33	3	99,49	91,99	99,37	3	99,49	91,99	99,73	2

To summarize the results, the estimators mean performance over all corpuses is taken and displayed in the tables below. One table for each for each λ. The tables are sorted after the mean rank $(\overline{R})$.

Table 14. Mean results for $\lambda = 1$

Overall Rank	$\lambda = 1$			
Est.	$\overline{p}$	$\overline{r}$	$\overline{W}_{Acc}$	$\overline{R}$
p_{SGT}	96,57	95,33	97,54	2,50
p_{WB}	96,26	95,40	97,49	3,00
p_{Rob}	97,07	94,05	97,34	3,00
p_{ELE}	94,99	94,60	96,89	3,67
p_{Lid}	96,93	92,89	97,03	3,83
p_{abs}	92,14	95,22	96,28	5,50
p_{lap}	91,72	94,85	96,02	5,67

Table 15. Mean results for $\lambda = 9$

Overall Rank	$\lambda = 9$			
Est.	$\bar{p}$	$\bar{r}$	$\overline{W}_{Acc}$	$\overline{R}$
p_{SGT}	96,87	94,94	98,55	2,67
p_{Lid}	97,31	92,21	98,54	2,83
p_{WB}	96,60	95,03	98,53	3,00
p_{Rob}	97,28	93,63	98,56	3,17
p_{ELE}	95,57	93,94	98,00	4,33
p_{abs}	92,59	94,74	97,09	5,83
p_{lap}	92,55	94,44	96,90	6,00

Table 16. Mean results for $\lambda = 99$

Overall Rank	$\lambda = 99$			
Est.	$\bar{p}$	$\bar{r}$	$\overline{W}_{Acc}$	$\overline{R}$
p_{Lid}	97,47	91,73	98,88	2,17
p_{SGT}	97,20	94,40	98,92	2,50
p_{Rob}	97,54	93,05	98,86	3,17
p_{WB}	96,88	94,76	98,81	3,33
p_{ELE}	96,03	93,34	98,34	4,33
p_{lap}	93,29	93,62	97,34	5,17
p_{abs}	93,28	94,16	97,57	6,33

6.2.3 Analysis

As expected, the overall performance of the Laplace estimator was poor except on PUA where it had the highest rank for $\lambda = 1$. The reason for this could be that PUA includes legitimate non-English messages in Greek. Rare Greek words are unlikely to occur in spam, hence moving a great deal of probability mass to rare events may increase the precision. The performance of the Absolute estimate was similar to that of Laplace and together they ranked bottom of all ranking lists. ELE performed slightly better than Laplace and Absolute and was ranked in the middle for all tests.

There was no outright winner outperforming all the other; however there are four candidates for the top spot. With regards to both high precision and recall, the Simple Good Turing and Witten Bell estimate performed very well on most corpuses.

What is surprising is the performance of Lidstone with $\delta = 0.1$ as it maintains high precision over all corpuses. It was expected it to behave more a like ELE and Laplace. In fact, it was ranked as the top smoother for $\lambda = 99$ (see Table 14).

Robinson's Bayesian estimate is very attractive as it maintains high precision in all tests. In the ranking table it achieves the highest average precision for $\lambda = 1$ and $\lambda = 99$.

6.2.4 Conclusion

Simple Good Turing has previously shown (Gale 1995) impressive performance which our tests strengthen. Perhaps the main obstacle with the Simple Good Turing lies in its implemental complexity. Our test results show that Witten Bell is a reasonable substitute for Good Turing with only a small performance loss. Witten Bell is simple to implement and well used in natural language. Even though Lidstone performed very well, it cannot be considered to be a robust estimate until it has gone through more evaluation. $\delta = 0.1$ was found through manual testing of different values and it is possible that it performs less well on different corpuses. Robinson's estimate, based on a Bayesian smoother showed excellent results, it is easy to implement and less computationally expensive than both Witten Bell and Good Turing.

Table 17. Overall performance of the tested estimators.

Estimator	Performance	Easy to implement	Need manual parameters
p_{SGT}	Excellent	No	No
p_{Rob}	Excellent	Yes	Yes
p_{Lid}	Excellent?	Yes	Yes
p_{WB}	Good	Quite	No
p_{ELE}	Average	Yes	No
p_{lap}	Poor	Yes	No
p_{abs}	Poor	Yes	No

6.2.5 Future work

All experiments were done on full corpuses; further research on smaller corpus sizes is encouraged to examine how they scale. More importantly, this work has only experimented with a few estimators and there are many other interesting smoothers such as Kneser-Ney smoothing (Nay & Kneser 1995), Katz smoothing (Katz 1987) and Church-Gale smoothing (Church & Gale 1991).

6.3 Feature selection

This experiment explores the feature selection performance as a function of the number of selected features. Since e-mails do not usually have many features it was interesting to investigate whether feature selection is necessary when using a Naive Bayesian spam classifier. This experiment compared feature selections time gain and accuracy influence against using all features.

6.3.1 Experimental settings

All tests were conducted with stratified 10-fold cross-validation. Due to the promising results of the Simple Good Turing estimator (see section 6.2.4) it was used throughout the tests. The performance was measured by the weighted accuracy using $\lambda = 9$. The same set of delimiters was used for all tests.

6.3.2 Results

The results are displayed in Diagram 1 to Diagram 8. A base-line with the performance of all features selected is presented in each diagram. The diagrams show how the performance $WAcc$ of the classifier varies as the number of features increases for the three tested feature selection methods.

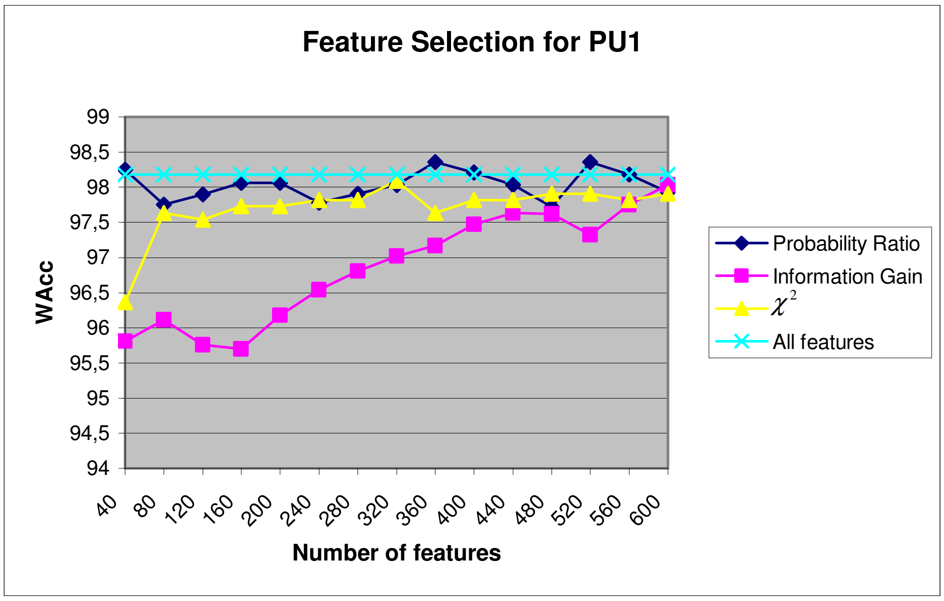

Diagram 3. Performance as a function of the number of features selected on PU1.

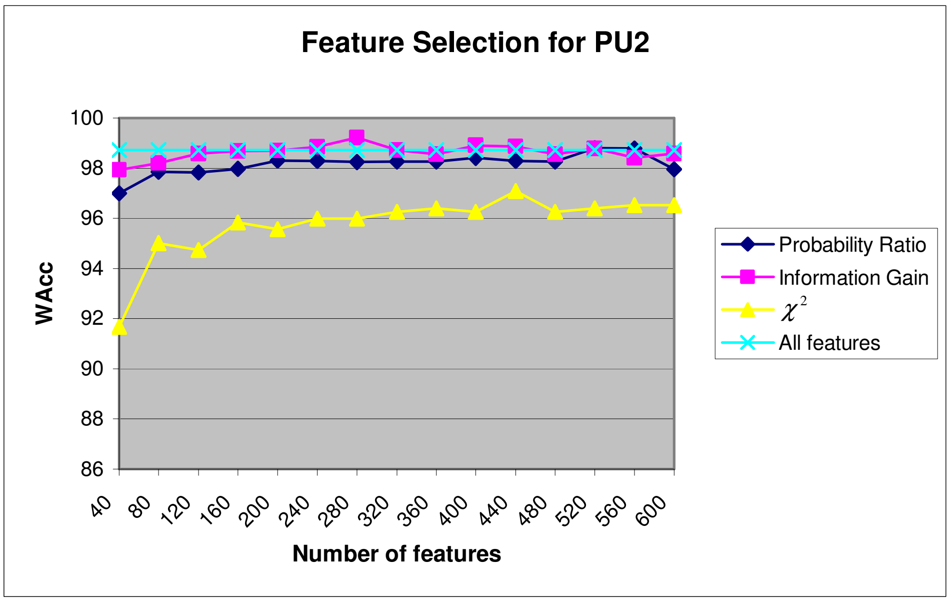

Diagram 4. Performance as a function of the number of features selected on PU2.

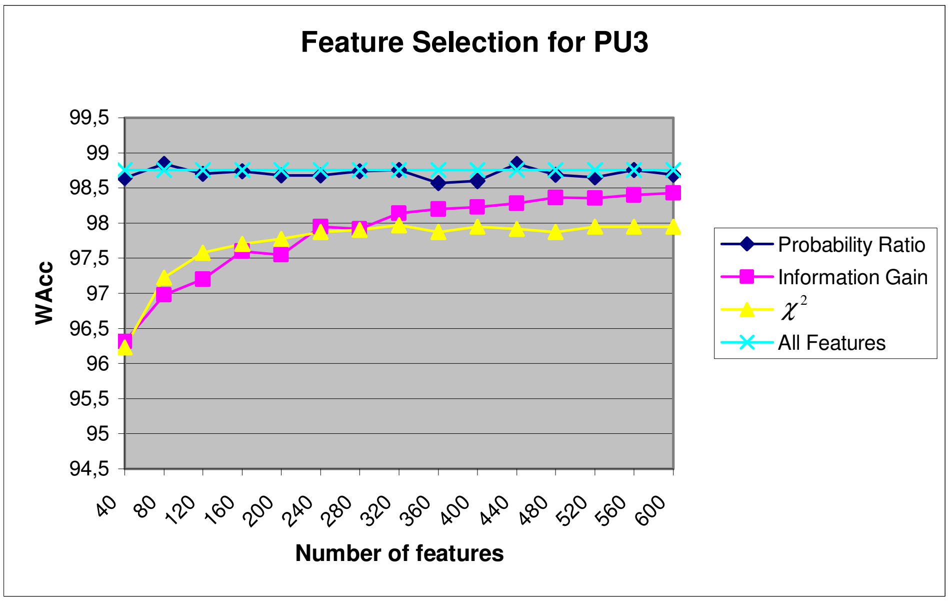

Diagram 5. Performance as a function of the number of features selected on PU3.

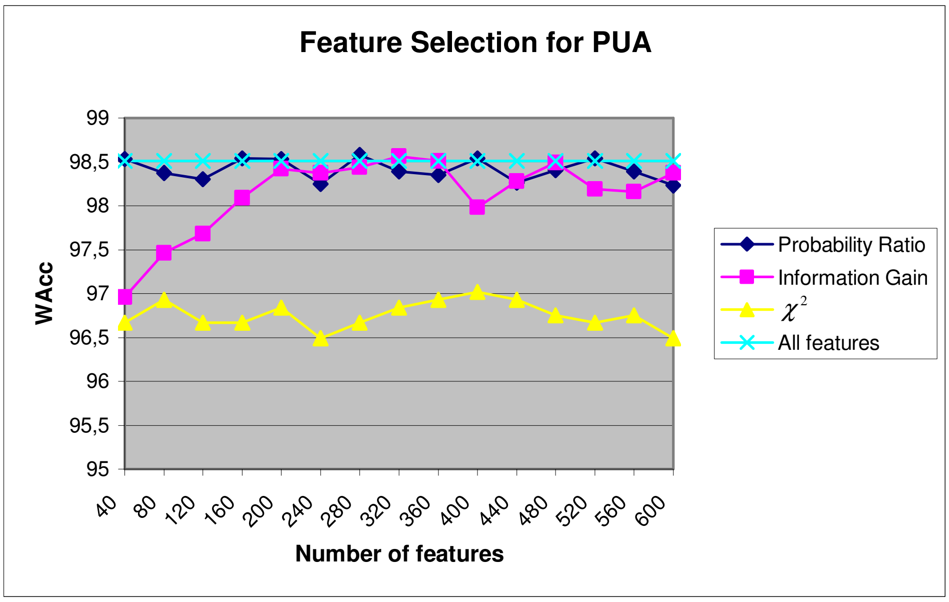

Diagram 6. Performance as a function of the number of features selected on PUA.

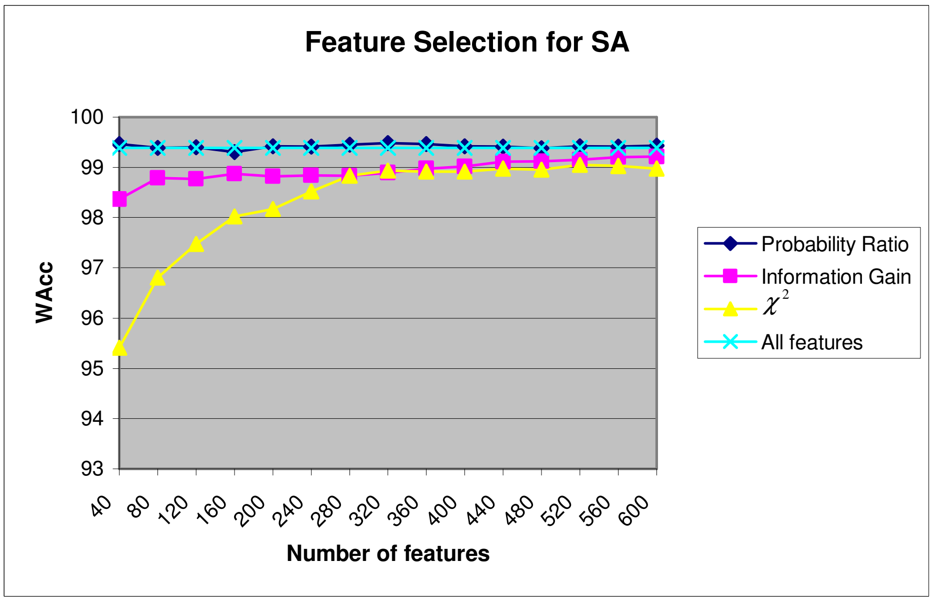

Diagram 7. Performance as a function of the number of features selected on SA.

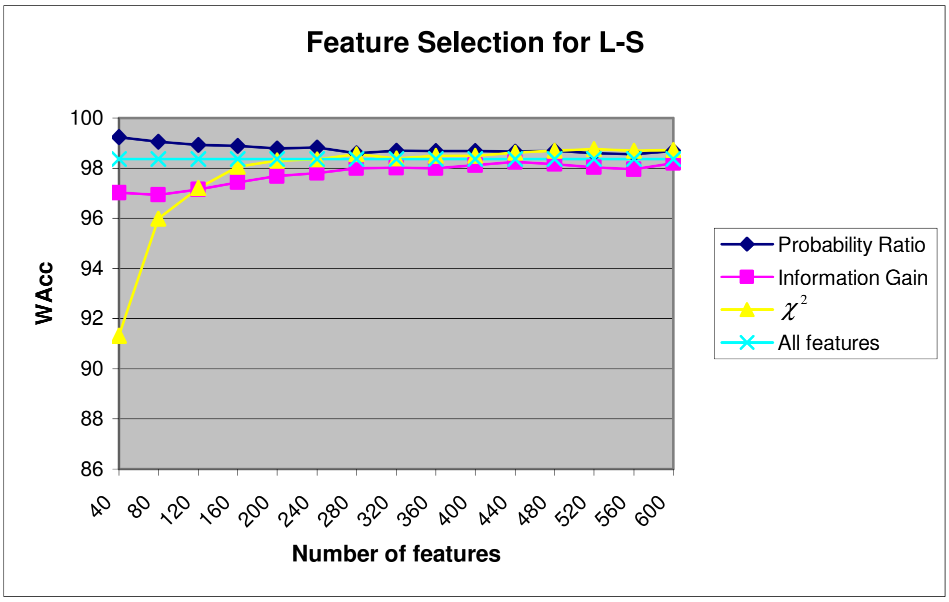

Diagram 8. Performance as a function of the number of features selected on L-S.

Diagram 9 shows how the classification time changes as the number of features increases.

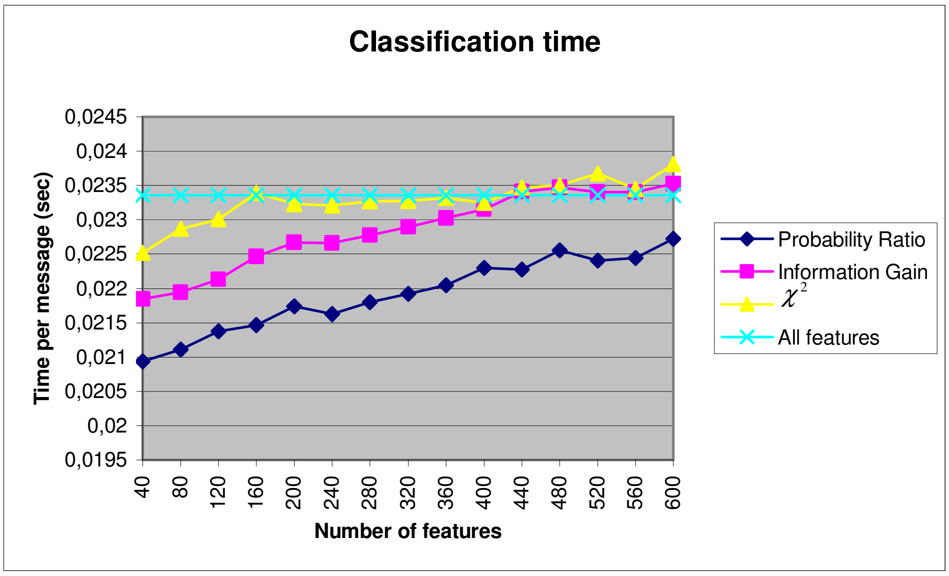

Diagram 9. The average time to classify one message for the different feature selectors.

6.3.3 Analysis

The test results validated those already found (Mladenic & Grobelnik 1999), the Probability Ratio (which is closely related to the Odds-Ratio) works very well with a Naive Bayesian classifier and did, in fact, outperform both Information Gain and χ^2 in all tests apart from PU2, where Information Gain was slightly better. The reason for this may be that the Simple Good Turing estimator is less accurate for PU2 (see Table 9) than the other corpuses.

Information Gain and χ^2 both favor common terms, while the Probability Ratio favors those with high probability of being either good or spam. This emphasizes the importance of rare words as indicators for classification. Of course a good estimator is required to rely on the probability ratio.

The diagrams also revealed that the features selected by the Probability Ratio roughly estimated the performance of using all features. For Ling-Spam it was even more accurate than using all features. The reason for this could be that Ling-Spam may produce over optimistic results (Androutsopoulos *et al*. 2004) due to its characteristics.

To classify a message using all features took on average (see Diagram 9)

$$t_{All} = 0.023$$

The maximum time gained by using feature selection was obtained by Probability Ratio. The average time when the number of features where reduced to 40 was

$$t_{PR_{40}} = 0.021$$

This means that approximately

$$t_{All} - t_{PR_{40}} = 0.023 - 0.021 = 0.002s = 2ms$$

is gained per message, i.e. a saving of about 10% of the total time per message.

6.3.4 Conclusion

The experiments with feature selection showed that the Probability Ratio was superior to both Information Gain and χ^2 when using a Naive Bayesian classifier in the spam domain.

In addition to this, this experiment attempted to answer the question as to whether feature selection was required in this domain, which is perhaps more important. It was shown that if Probability Ratio is used for feature selection, it is possible to improve the classification time by 10% without any significant accuracy loss. In this (non-optimized) test environment the time saved per message was about $2ms$. This effectively means that for every thousand e-mails, 2 seconds are saved. This time gain was considered to be far too small to encourage the use of feature selection as a time saver. Furthermore, none of the tested feature selection methods consistently improved the performance over using all features. Since none of the tested feature selection methods improves performance or saves time, the recommendation is to use all features, until a better solution is found.

6.3.5 Future work

This experiment has concluded that the primary purpose of feature selection for Naive Bayesian spam classification is not to reduce the feature dimensionality. The basic purpose is to improve the performance when using all features. Such a feature selection algorithm was presented in (Forman 2003) under the name BNS (bi-normal separation). The author conducted some preliminary testing with BNS on the PU corpuses and it did perform approximately as the Probability Ratio. However, further more controlled experiments should be carried out on BNS and other interesting feature selection methods.

7 Summary

This work has empirically tested the tokenizer, estimator and feature selection for a Naive Bayesian spam classifier.

It was shown that a good delimiter subset is likely to be of low cardinality. The reason is due to the information loss of larger subsets.

Further it was found that Simple Good Turing and Robinson's Bayesian smoother are good choices for estimating probabilities.

Finally this work showed that feature selection is not required to decrease the already low dimensionality of features in messages. Hence, a feature selection method should only be used if it benefits the performance. In these tests both Information Gain and χ^2, which are common for selecting features for a Naive Bayeisan classifier, degraded the performance of the classifier. The Probability Ratio outperformed both Information Gain and χ^2 but did not consistently increase the classifier performance. The recommendation is thus to keep all features and only use feature selection if it improves the performance over a majority of test corpuses.

To describe the outcome of this work in one sentence

Use space as delimiter (and possibly a few more), estimate the probabilities with Simple Good Turing or Robinsons' Bayesian smoother and do not use any feature selection.

References

Almuallim, H. and T. Dietterich. (1991), Learning with many irrelevant features. *In Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 547-552. Menlo Park, CA: AAAI Press/The MIT Press.

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

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

Androutsopoulos, I., Paliouras, G., Michelakis, E. (2004), Learning to Filter Unsolicited Commercial E-Mail. Athens University of Economics and Business and National Centre for Scientific Research "Demokritos"

Bevilacqua-Linn M. (2003), Machine Learning for Naive Bayesian Spam Filter Tokenization

Breiman, L., and Spector, P. (1992), Submodel selection and evaluation in regression: The X-random case. *International Statistical Review*, 60, 291-319.

Bullard, F. (2001), A Brief Introduction to Bayesian Statistics, http://courses.ncssm.edu/math/TALKS/PDFS/BullardNCTM2001.pdf, 2004-03-13.

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

Church, K.W., Gale, W.A. (1991), A comparison of the enhanced GoodTuring and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19--54.

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

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

Ferri F.J., Pudil P., Hatef, M. and Kittler J. (1994), Comparative Study of Techniques for Large-Scale Feature Selection. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice IV*, pages 403-413. Elsevier.

Gale, W. A, Church, K. W. (1994), What is wrong with adding one? In Oostdijk, N. and de Haan, P. (eds) *Corpus-Based Research into Language*, 189-198. Amsterdam: Rodopi.

Gale, W. A. (1995), Good-Turing Smoothing without Tears. *Journal of Quantitative Linguistics*, pp. 217-254.

Good, I.J. (1953), The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237-264.

Graham, P. (2002), A plan for spam. http://www.paulgraham.com/spam.html, 2003-11-13

Heckerman, D. (1995), A Tutorial on Learning Bayeisan Networks, http://citeseer.ist.psu.edu/135897.html, 2004-03-13

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

Holden, S. (2003), Spam Filters, http://freshmeat.net/articles/view/964/, 2004-03-18.

Jain, A.K. and Zongker, D. (1997), Feature Selection: Evaluation, application, and small sample performance. IEEE Trans, *PAMI* 19. pp. 153-158.

Jain, A.K. and Mao, J. (1999), Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, (2000) 4-37.

Katz, S. M. (1987), Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics*, Speech and Signal Processing, ASSP-35(3):400--401, March.

Kira, K., and Rendell, L.A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *AAAI-92, Proceedings of the Ninth National Conference on Artificial Intelligence,* pp. 129-134. AAAI Press.

Kohavi, R. (1995), A study of cross-validation and bootstrap for accuracy estimation and model selection, *International Joint Conference on Artificial Intelligence (IJCAI).*

Kohavi, R., John, G.H. (1996), Wrappers for feature subset selection. Artificial Intelligence

Kushmerick, N. (1997), Wrapper induction for information extraction, Ph.D. Thesis, University of Washington, Seattle, WA.

Kushmerick, N. (1999), Wrapper induction: Efficiency and expressiveness, Artificial Intelligence 118.

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

Mertz, D., (2002), Spam filtering techniques, http://www-106.ibm.com/developerworks/linux/library/l-spamf.html#author1, 2004-02-03.

Mladenic D. and Grobelnik M. (1999), Feature Selection for Unbalanced Class Distribution and Naive Bayes. *In Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 258-267.

Nay. H., Essen, U. and Kneser, R. (1994), On structuring probabilistic dependences in stochastic language modeling. *Computer, Speech, and Language*, 8:1-38.

Nay. H. and Kneser, R. (1995), Improved backingoff for mgram language modeling. *In Proceedings of the IEEE International Conference on Acoustics*, Speech and Signal Processing, volume 1, pages 181--184.

Ng A.Y. and Jordan M.I. (2002) On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *In T. Dietterich, S. Becker and Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems* (NIPS) 14.

Pudil, P. and Kittler J. (1994), Floating search methods in feature selection, *Pattern Recognition Letters*, vol 15, no. 11, pp. 1119-1125.

Rijsbergen, C. J. (1979), ”Information Retireval. Butterworths”, London.

Robinson, G., (2003), Spam Detection, http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html, 2004-01-13.

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

Siedlecki, W. and Sklansky, J. (1988), On automatic feature selection, *International Journal of Pattern Recognition and Artificial Intelligence* 197-200.

Somol, P., Pudil, P., Novoviˇcov´a, J. and Pacl´ık, P. (1999), Adaptive floating search methods in feature selection, *Patt. Recog. Lett.*, 20(11–13):1157–1163.

Spance C. and Sajda P. (1998), The role of feature selection in building pattern recognizers for computer aided diagnosis. National Information Display Laboratory, Sarno Corporation, Princeton.

Stanley, F.C. and Goodman, J. (1996), An Empirical Study of Smoothing Techniques for Language Modeling. *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310-318.

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

Vapnik V. (1995), The nature of statistical learning theory. *Springer*, New York.

Witten, Ian H. and Timothy C. Bell. (1991), The zerofrequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085-1094, July.

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

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