Feature Selection Strategies for Spam E-mail Filtering

Ren Wang

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical Engineering) at

Concordia University

Montreal, Quebec, Canada

April 2006

Your file Votre référence
ISBN: 978-0-494-20756-7
Our file Notre référence
ISBN: 978-0-494-20756-7

Abstract

Feature Selection Strategies for Spam E-mail Filtering

Ren Wang

The spam e-mail (also known as junk e-mail) problem is rapidly becoming unmanageable. According to a recent European Union study, junk e-mails cost all of us about 9.4 billion (US) dollars per year, and many major ISPs say that spam adds about 20% to the cost of their service.

Feature selection is an important research problem in different text categorization applications including spam e-mail filtering. In designing spam filters, we often represent the e-mail by vector space model (VSM) in which every e-mail is considered as a vector of word terms. Since there are many different terms in the e-mail, and not all classifiers can handle such a high dimension, only the most powerful discriminatory terms should be considered. Also, some of these features may not be influential and might carry redundant information which may confuse the classifier. Thus, feature selection, and hence dimensionality reduction, is a crucial step to get the best out of the constructed features.

Many feature selection strategies (FSS) can be applied to produce the desired feature set. In this thesis, we investigate the use of several classifier-dependent feature selection strategies. We cast our feature selection problem as a 0-1 optimization problem and different optimization techniques are compared. These techniques include several local search optimization algorithms such as Hill Climbing, Simulated Annealing, Threshold Accepting and Tabu Search. We also examine some other algorithms inspired by

biological systems and artificial life techniques such as Genetic Algorithm, Particle Swarm Optimization, Ant Colony Optimization and Artificial Immune Systems. The performance of all the above algorithms is compared with some traditional dimensionality reduction techniques such as Principle Component Analysis, Linear Discriminant Analysis and Singular Value Decomposition.

Our experimental results show that all these techniques can be used not only to reduce the dimensions of the e-mail VSM, but also improve the performance of the spam filter.

Acknowledgments

My heartfelt thanks and earnest gratitude are due to Dr. Amr Youssef and Dr. Ahmed Elhakeem for their constant support and guidance and for their enduring patience. Their scientific skills, generosity, enthusiasm and support are unmatched.

I'd like also to express my appreciation to all the faculty and people at Concordia Institute for Information Systems Engineering who contributed to my success one way or the other.

My deepest love and supreme appreciation and gratitude to my parents, Guangyu and Changying, for their unceasing love and relentless patience and devotion raising me. My thanks go to my brother Zhu as well, for his love and encouragement.

Thank you to all my friends and lab mates, especially Kun Liu, Ziad Saber, Mohammad Faisal Uddin, Ying Zhang, Yongxing Hu and Yong Fu whom I can't thank enough for their continual encouragement and unvarying support and assistance, and for their true friendship and loyalty through the thick and thin.

To my wife Wenlan that she was so patient and for giving me her love, tender and caring.

Finally, to all the people who impacted my life positively: *Thank you*!

Table of Contents

List of Figures

List of Tables

List of Notations

TF-IDF	Term frequency and Inverse document frequency
FSS	Feature Selection Strategies
KNN	K-Nearest Neighboring
NB	Naive Bayes
PCA	Principle Component Analysis
SVD	Singular Value Decomposition
LDA	Linear Discriminant Analysis
SC	Spectral Clustering
HC	Hill Climbing
TA	Threshold Accepting
TS	Tabu Search
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
GA	Genetic Algorithm
AIS	Artificial Immune Systems

Chapter 1

Introduction to Spam E-mail

1.1 Introduction

Electronic mail (e-mail) has become extremely important in our daily life because of its high speed and low cost. People are receiving an increasing amount of e-mail both on the job as well as for personal communications. On the other hand, we also receive many e-mails from some strangers we do not know. Most of these e-mails are commercial advertisement useless to the majority of us and sometimes they are harmful messages containing viruses or malicious codes. These e-mails are called Junk e-mail or Spam.

Junk e-mail is the electronic equivalent of junk paper-mail: unsolicited, and usually of a commercial/advertising nature. In Internet terminology, we may also come across junk e-mail described as UCE (Unsolicited Commercial e-mail), UBE (Unsolicited Bulk e-mail) or *Spam*. Many people regard any unsolicited e-mail as junk, irrespective of its source or contents.

Junk e-mails exist because vast amounts of e-mail can be sent for very low cost and the sender will get some profits from such e-mails. This is in part due to the shared-cost

structure of the internet, which has each end of a communication pay the costs of their end, but in truth, it is primarily because e-mail is a very efficient and very cheap communications technology. For one to one communications, there has never been anything anywhere near as cheap, which is part of the reason people like it.

1.2 A Brief History of Spam

When the internet first began, there was no such thing as spam. There was plenty of e-mail, but none of it was of a commercial nature. The internet began as a military and educational project. Making money had nothing to do with anything, so there was no reason to send out commercial mailings.

Probably the first spam was written by an employee at Digital Equipment Corporation [1]. It was intended to be sent to every e-mail address on the ARPANET, but since space was limited the later names were truncated.

One of the first truly despicable spam messages came from Dave Rhodes [1]. Rhodes wrote an e-mail advertising a pyramid scheme. This spam was not really an e-mail message as it was posted to the Usenet (newsgroups) but the concept was the same. Lots of people got to read an e-mail advertising a silly scheme with a subject of "MAKE MONEY FAST!!"

In 1993, a man with the name of Richard Depew [1] decided to introduce a concept known as retro-moderation. This would allow newsgroups to become a little bit more controlled, by having a moderator who would cancel postings after they had been made. While there were moderated newsgroups up to this point, Depew was suggesting moderating after the fact. Depew wrote a program to delete these postings. Unfortunately,

it had a bug and wrote 200 messages to the *news.admin.policy* newsgroup. This annoyed a lot of people, who, for the first time, called the messages *spam*. This is the first known time that this kind of thing was referred to as spam.

One of the first mass mailings on the Usenet was from Clarence L. Thomas IV [1]. The subject was "Global Alert for All: Jesus is Coming Soon" and it was a long, boring e-mail about the end of the world. This mailing occurred in January of 1994.

Spam in the modern sense began in 1994 when two men named Cantor and Siegel [1] posted an advertisement for "Green Card Lottery". They posted this message to 6,000 newsgroups at the same time. They continued posting for some time, and reportedly made some money from their efforts.

1.3 The problems of Junk e-mail

The junk e-mail problem is rapidly becoming unmanageable, and threatens to destroy e-mail as a useful means of communication.

We waste increasing amounts of our time deleting junk e-mails and sorting out these e-mails from our e-mail box every day, otherwise junk e-mails will congest the e-mail box and we can not receive legitimate e-mails anymore. We also have to pay the cost of such messages. People pay for an e-mail mailbox for various reasons, but not because they want to receive advertising. It costs the recipient real money in terms of extra connect-time charges, phone time charges, disk space, and lowered bandwidth. Some kinds of spam are illegal in some countries. Especially with pornography, mere possession of such material can be enough to put the recipient in jail.

Secondly, junk e-mails will occupy server storage space and consumes network bandwidth. The mail server is one of the busiest servers in the enterprise network or ISP. It receives many requests and delivery many e-mails every day. So the network recourses and performance are very important for mail servers. But with junk e-mails, all the recourses will be consumed quickly and they even block the whole network and shut down the mail servers.

Finally, there is also the fear that such junk e-mails could hide viruses, which then infect the whole network. Many viruses can be sent to everybody in the network within the junk e-mails. Such e-mails with virus will infect the user's computer even the whole network if the e-mail receivers open the e-mails and run the virus programs by accident.

1.4 Anti-Spam Approaches

Many approaches can be applied to protect the e-mails system from junk e-mails. However, none of the currently known approaches effectively copes with the huge volume of traffic with sufficient accuracy. In practice, we often combine several techniques together to solve this problem. We can divide these techniques into three main catalogues: network level techniques, user level techniques and policy based techniques. Figure 1.1 shows the three catalogues with some examples.

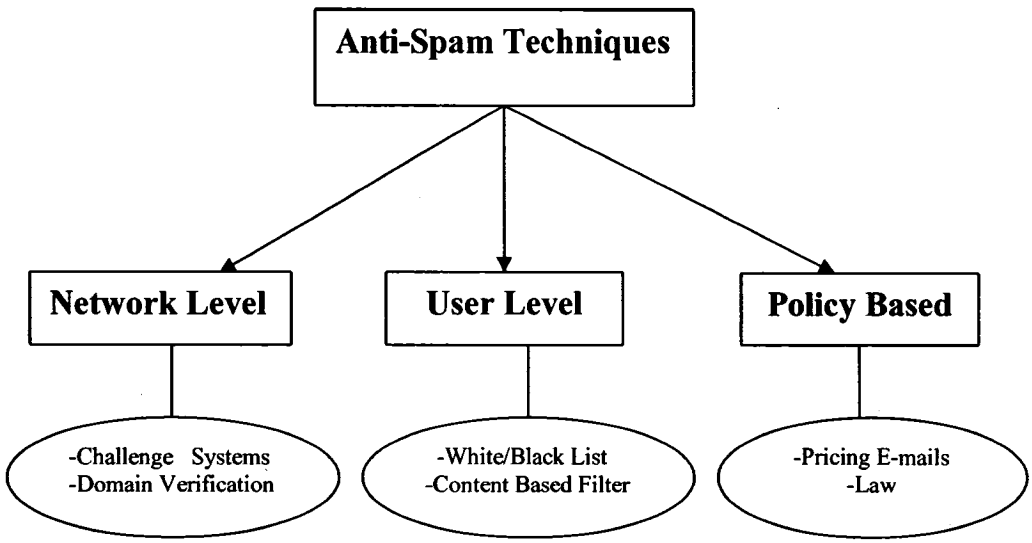

Figure 1.1: General classification of anti-spam techniques.

1.4.1 Network level Techniques

Because each e-mail will be sent via network systems, one may try to develop some approaches based on the network to trace the junk e-mail and find out the spammers.

1.4.1.1 Domain Verification

One of the most difficult problems in dealing with spam is that we can not decide accurately, who sent the message. The ability to trace e-mail reliably to a specific sender, a specific sending domain, or a specific route through the simple mail transfer protocol (SMTP) infrastructure would increase the effectiveness of many classification mechanisms, and would also increase the feasibility and effectiveness of legal remedies.

Domain verification schemes, of which there are several proposals currently outstanding, including SPF, Caller ID, and Domain Keys, all attempt the same basic task [2]: choose an indication in the message of who the "sender" is (which choice varies

among the schemes), get information directly from that sender's domain that helps verify that this message really is coming from that domain, and use that as input to the decision about how to handle this message. In theory, this means that a sender outside of "ABC" could not send mail that appeared to come from ABC.com, without having that mail flagged as suspicious. The recipient's filtering system can then use the results of this verification ("the domain is verified", "the verification failed", or "the sending domain does not provide data"), along with some "reputation" of the sending domain, as a factor in its filtering decision.

1.4.1.2 Challenge/Response Systems

A challenge/response system is a program that replies to an e-mail message from an unknown sender by subjecting the sender to a test designed to different humans from automated senders. The system ensures that messages from people can get through and the automated mass mailings of spammers will be rejected. Once a sender has passed the test, the sender is added to the recipient's *White list* of permitted senders that won't have to prove themselves each time they send a message.

Challenge/response systems take a number of different approaches to the task of separating humans from machines. Typically, when a message is received, the system sends a reply that includes a URL linking the user to a Web site. At the Web site, the user is asked to perform some task that, while easy for a human, is beyond the capabilities of an automated spamming program. The system might ask the answer to a simple question, for example, or require the user to copy distorted letters or numbers displayed in an image.

In the absence of adequate authentication mechanisms, challenge/response system is a way to ensure that this message is not part of a mass-mailing. To be effective, a challenge/response system must have a sufficient variety of challenges to provide a strong defense against trial-and-error attacks. On the other side, such systems may cause problems in certain contexts [2]. Recipients of business messages, particularly those who will often hear from new or potential customers, may find that the challenges annoy correspondents and result in lost business. Someone doing one a favor might not respond to a challenge. One should also note that there are reasons for an automaton to send e-mail – not all of it is spam.

These effects can be mitigated by combining this technique with others. A challenge might only be sent once a message is determined, by other means, likely to be spam. If the classification is uncertain, or if the recipient's preferences specify it, a challenge may then be sent. The problem of having challenges sent to unwitting third parties, whose e-mail addresses have been "spoofed" in spam, may at least partially be resolved with the domain verification mechanism: the challenge is not sent if the domain fails to verify.

1.4.1.3 Other Examples for Network Based Techniques

There are still some other approaches based on the network. Kang *et al.* [3] introduced a framework named TCP damping that slows down the spammers by carefully tuning e-mail delivery parameters, without changing the current e-mail delivery software. The anti-spam mechanisms under this framework directly add delay to the e-mail delivery process, and increase computation cost at the sender side for delivering e-mails. Within the anti-spam framework, [3] use existing spam filters to generate a spam likelihood value for each message at its delivery time. The likelihood information is used

by the recipient to drive the spam resistance mechanisms that control the delay and cost of e-mail delivery. The mechanisms of adding delay and cost are implemented at the TCP level. TCP is the transport protocol used by the Simple Mail Transfer Protocol (SMTP), which is the protocol used by Internet e-mail. Slowing a TCP connection from the receiver side can be achieved in many ways, such as postponing TCP acknowledgements or injecting congestion notification bits to spoof congestion. To increase sender computation cost, the receiver could force the sender to generate more packets to deliver the same message. Since all e-mails are delivered over TCP, selectively delaying or adding cost to a suspicious TCP sender would effectively slow down spammers. This spam resistance effect is achieved based on the assumption that the TCP connections carrying spam are more likely to be identified as suspicious connections by many users, whereas the TCP connections for normal e-mails are unlikely to be misclassified by a large number of users simultaneously.

1.4.2 Policy Based Techniques

Although some people choose to reduce the junk e-mails they are receiving by using some technical tools, others may decide to fight back with spammers using other approaches.

Existing examples of such anti-spam approaches include pricing e-mail and accuse spammers by law. It increases the cost of delivering a message by requiring real money from the sender and against the spammers based on the law.

1.4.2.1 Pricing E-mails

The financial truth that enables spam in the first place is that sending spam costs so little that even a negligible response rate makes it worth the expense of sending. It is a big

social change to make, but a system of collecting payment for sending e-mail would turn the financial model around, and would make an enormous difference in the amount of spam sent, and in the content of it. Even a charging a small fraction of what is charged for paper mail would make a large difference in how spam is paid for. Much of the debate over postage schemes centers on who collects the postage. One scheme called charity seals would let each user directs postage to charities of the user's choice [2]. Another, known as attention bonds, lets each recipient set a price for which an unknown sender can post a bond to get e-mail through; for wanted e-mail, the bond is returned, but for spam it is kept by the recipient as payment for time wasted. Two of the world's biggest e-mail account providers, Yahoo Inc. and America Online, plan to introduce a service that would charge senders a fee to route their e-mail directly to a user's mailbox without first passing through junk mail filters. The fees, which would range from 1/4 cent to 1 cent per e-mail, are the latest attempts by the companies to weed out junk e-mail. In exchange for paying, e-mail senders will be guaranteed their messages will not be filtered and will bear a seal alerting recipients they are legitimate. But we still consider it unrealistic to expect payment systems to be part of the solution in the short term.

1.4.2.2 Law System

Another weapon in our anti-spam arsenal is the legal system, and the law is beginning to be used in fighting the spammers. In recent cases in New York [4] and Florida [5], spammers were sued under older fraud statutes, while a consortium of ISPs filed six spam lawsuits in USA federal courts in four states against a number of spammers [6]. On January 2006, America Online has won more than $5 million in a judgment against a Minnesota spammer under the federal CAN-SPAM act. Spammer

Christopher William Smith is believed to be one of the world's most notorious spammers. He was accused of sending billions of spam messages to AOL account holders in 2003, promoting fake college degrees, cable descramblers, generic impotence drugs, and explicit Web sites. AOL first sued Smith in 2003, but refiled again in 2005 to take advantage of 2004's CAN-SPAM federal legislation and this case helped spawn the federal legislation. Some of the techniques already described, particularly those that validate the sender and the message routing, can help support legal action, providing evidence that may lead to convictions.

1.4.3 User Level Techniques

The popular means for solving the spam problem is to deploy an e-mail filter to classify the spam and legitimate e-mails automatically at the user side. Other simplistic approaches include black-listing and white-listing. In practice, effective spam filtering uses a combination of these three techniques.

1.4.3.1 White/Black List

Two of the least effective methods for fighting spam are *white* lists and *black* lists. A spam black list is a list of IP addresses and domains of known spam e-mail servers. Black lists are used to block all e-mail that comes from certain servers on the Internet that has been identified as being used to send spam. A well-known black list is hosted by SpamCop, located at *www.spamcop.net*. Another one is Open Relay Database, located at *www.ordb.org*. Many anti-spam products also maintain their own black lists and include optional subscriptions to third-party black list services.

White lists are the opposite of black lists. They list trusted e-mail addresses and domains that are always allowed to send e-mail, no matter what the content is. White lists

are used to require that senders authenticate their identity prior to e-mail being delivered to the recipient. White lists will definitely allow e-mail coming from a trusted site to come through, but do not provide a solution for blocking spam. White lists require constant maintenance to be very effective. If not properly maintained, the risk of losing e-mail from legitimate sources is high.

1.4.3.2 Content Based Filter

The heart of any anti-spam system is the part that classifies messages, and filters them. It is the filters that determine which mail should get through and which should be considered spam, and a good set of filters provides a great deal of flexibility and customization. The simplest and most common approaches are to use filters that screen messages based upon the presence of words or phrases common to junk e-mail, but the goal of all classifiers is to give the e-mail a score that can be used as input to filters. Classifiers may analyze the e-mail's headers, the body of the e-mail, its structure, or a combination of all. They may consider only this one e-mail, or may compare it to other e-mails. They may give a numeric score that indicates how likely they determine this e-mail to be spam, or they may attach a set of keywords or categories, highlighting aspects of the message that might be considered by filters.

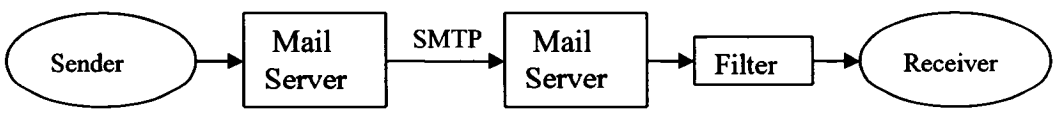

Figure 1.2: General model for e-mail filter system.

But until today, the spam filters are less than perfect protection to our e-mail system. Its less discriminating has the problem of false positives; they mistakenly block or misclassify legitimate messages as spam. The damage of a single false positive can be very serious, depending on the e-mail content. Because of its unsatisfied accuracy, we still receive some junk e-mail also. So how to improve the accuracy of the filter is a big challenge to the researchers. Many people focus on developing different algorithms based on the e-mails content, others try to analyze the content of junk e-mails and legitimate e-mails.

Early spam filters were instances of knowledge engineering using hand-crafted rules. The process of creating the rule base requires both knowledge and time, and the rules were thus often supplied by the developers of the filter. Recently, a shift towards a more statistical approach has occurred as more focus has been put on machine learning for the automatic creation of personalized spam filters. A supervised learning algorithm is presented with e-mails from the user's mailbox and outputs a filter. The e-mails have previously been classified manually as spam or non-spam. The resulting spam filter has the advantage of being optimized for the e-mail distribution of the individual user. It is able to use also the characteristics of non-spam, or legitimate e-mail during classification.

From the machine-learning viewpoint, spam filtering based on the textual content of e-mail can be viewed as a special case of text categorization, with the categories being spam or non-spam. In text categorization, the text can be represented by vector space model (VSM). Each e-mail can be transferred into the vector space model. This means every e-mail is considered as a vector of word terms. Since there are many different words in the e-mail and not all classifiers can handle such a high dimension, we should

choose only the most powerful discriminatory terms from the e-mail terms. Another reason is some of these features may not be influential and might carry redundant information which may confuse the classifier. Thus, feature selection, and hence dimensionality reduction, is a crucial step to get the best out of the constructed features and improve the performance of the filter.

1.5 Outline of the Thesis

The rest of this thesis is organized as follows. In chapter two, we briefly review some of the background related to content based filters and some other necessary definitions required throughout the thesis.

Chapter three gives a review of some traditional dimensionality reduction techniques for spam e-mails. We also report our experimental results based on these techniques as a baseline to compare with our newly proposed approaches.

In chapter four, we introduce some new classifier dependent features selection algorithms for e-mails filters and provide some experimental results for these new techniques.

Finally, chapter five provides a summary of our results and some directions for the future work.

Chapter 2

Content Based E-mail Filtering

2.1 Introduction

To accommodate the growing need for spam classifiers, a variety of techniques have been developed and applied towards the problem, both at the network and user levels. The simplest and most common approaches use filters that screen messages based upon the presence of words or phrases common to junk e-mail. So spam filtering can be thought of as text categorization task where the classes to be predicted are spam and legitimate. Most of the early spam filters were instances of knowledge engineering using hand-crafted rules (e.g. the presence of the string "advertisement money" indicates spam). The process of creating the rule base requires both knowledge and time, and the rules were thus often supplied by the developers of the filter. This is appropriate when few machine-readable texts were available and the computational power was expensive. Recent trends in the text categorization community have shifted to building classifiers automatically by applying some machine-learning algorithms to a set of pre-classified documents (training data). This is also called the statistical approach, in the sense that

differences among documents are usually expressed statistically as the likelihood of certain events, rather than some heuristic rules written by human. This trend is reflected in the goal of statistical spam filtering, which aims at building effective spam filters automatically from e-mail corpus. A variety of supervised machine-learning algorithms have been successfully applied to mail filtering task.

In this chapter, we begin with a formal definition of automated spam filtering task, followed by a brief review for some spam filtering algorithms based on the features or words in e-mails. We also analyze the header and body of e-mails which can improve the accuracy of these algorithms.

2.2 Spam Filtering As Text Categorization

2.2.1 Definition for Spam Filtering

First, we give a definition of automated spam filtering followed by [7].

Given e-mail dataset $D = \{d_1, d_2 ..., d_j, ..., d_{|D|}\}$ and category set $C = \{c_1 = spam, c_2 = legitimate\}$, where d_j is the j^{th} e-mail in D and C is the possible label set. The aim of automated spam filtering is to build a Boolean categorization function $\partial\ (d_j, c_i) : D \times C \rightarrow \{True, False\}$. When $\partial\ (d_j, c_i)$ is True, it indicates e-mail d_j belongs to category c_i; and when $\partial\ (d_j, c_i)$ is False, it means d_j does not belong to c_i.

For spam filtering there are two category labels: spam and legitimate. Each e-mail d_j can only be assigned to one of them, but not both. Therefore, we can use a simplified categorization function $W_{spam}(d_j) : D \rightarrow \{True, False\}$ instead. A e-mail is classified as spam when $W_{spam}(d_j)$ is True, otherwise it is legitimate e-mail.

Using the above notation, applying supervised machine-learning algorithms to spam filtering consists of two stages: the training stage and the classification stage.

During the training stage, a set of labeled e-mails must be provided as training dataset, which are first transformed into a representation that can be understood by the learning algorithms. The most commonly used representation for spam filtering is vector space model (VSM). Then we can run a learning algorithm over the training data to create a classifier $W_{spam}(d_j) \rightarrow \{True, False\}$.

During the classification stage, the classifier $W_{spam}(d_j)$ is applied to the vector representation of a new document d to produce a prediction whether d is spam or not.

2.2.2 Transfer E-mails to Vector Space

We first transform e-mail dataset into a representation that can be understood by the learning algorithms. There are many ways to represent e-mails, but the most commonly used representation for spam filtering is vector space model (VSM). The text can be represented by vector space model $tf = (tf_1, tf_2, ..., tf_n)$ where tf_i is the frequency of the i^{th} feature (term) in the e-mail. It is clear that terms appearing frequently in many e-mails have limited discrimination power (e.g., "the", "is", etc). We use Term frequency and Inverse document frequency (TF-IDF) representation to represent the e-mail. There are some different TF-IDF formulas and we choose following formula [8]

$$W(t,\vec{d}) = \frac{tf(t,\vec{d}) \times \log(N/n_t + 0.01)}{\sqrt{\sum_{t \in \vec{d}} (tf(t,\vec{d}) \times \log(N/n_t + 0.01))^2}}$$

where $W(t,\vec{d})$ is the weight of term t in e-mail $\vec{d}$ where $tf(t,\vec{d})$ is the frequency of the t^{th} term in e-mail d, N is the total number of e-mails in the corpus, n_i is the number of e-

mails that contain the i^{th} term and the denominator is the normalization factor. By this transformation, each e-mail can be represented as a vector. Accordingly, the more often a term appears in the e-mail, the more important this word is for that e-mail. On the other hand, the more often the word appears in the e-mail collection, the less informative this word becomes. In our experiments, we sorted the features by its document frequency (DF), i.e., the number of e-mails that contain the i^{th} features to choose 100 features for which DF range between 0.02 to 0.5 [10]. Thus the input to the feature selection algorithm is a feature vector of length 100. We then applied the feature selection algorithms to find out the most powerful discriminatory terms from the 100 features and test the performance of the e-mail filter.

2.2.3 Word Stemming and Stop-word

Word stemming and stop-word removals are two other issues that need to be considered before we transform the e-mails into vector space.

In English words having the same stem are usually assumed to have similar meaning, such as "walking", "walk", "walked". Stemming is performed to remove the affix and successor. Words which are very frequent and do not carry meaning (such as "a", "the") are called stop-words. These words are assumed not to carry any important information and so are usually ignored.

The main advantages of applying word stemming and stop-word removal are the reduction of feature space dimension and possible improvement on classifiers' prediction accuracy by alleviating the data sparseness problem. Androutsopoulos *et al.* [11] have investigated the use of word stemming and stop-word list on the performance of a naive Bayes classifier. Their result shows that most of time stemming and stop-word removals

do not have a statistically significant improvement over the nonstemming, no-stop-word removal filter. Furthermore, word stemming only applies to certain Latin-like languages, and is nonsensical to Asian languages such as Chinese or Japanese. But apparently, such process will reduce the feature space dimension.

2.2.4 E-mail Header and Body

Each mail consists of a header part followed by a body part and several optional attachment parts. Previous research mainly focused on building "pure" content-based filtering model that only uses features from message's subject and body parts. Now, some researches intend to figure out which parts of an e-mail have critical influence on the classification results. The header contains many fields, for example, trace information about which a message has passed (Received :), where the sender wants replies to go (Reply-To:), ID of this message (Message-ID:), format of content (Content-Type :), etc. Figures 2.1 illustrate the header of an e-mail. In [12], Zhang *et al.* show some particular features in the header field of a mail can give strong evidence whether a mail is spam or not. For instance, spam mails are seldom sent through normal e-mail client such as Outlook Express. Instead, spammers prefer to use some group mail sending software specially designed for dispatching mails to a large amount of recipients. This can be detected by looking at the X-mailer field in the header. If a mail has an X-mailer field indicating some group sending software or does not have X-mailer field at all, it is very likely to be a spam.

Zhang *et al.* [12] also show that the combined feature sets yielded a consistent improvement over feature sets that used either header or body feature alone. Therefore e-

mail headers are as important as mail bodies in terms of spam-discriminating power and should not be ignored by spam filter designers.

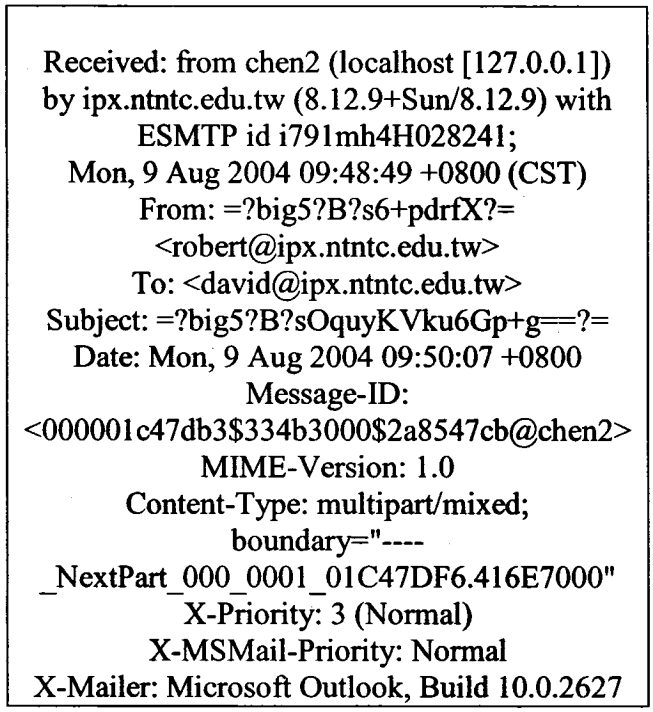

Figure 2.1: The e-mail header.

2.3 Performance Measure for Spam Filters

There are many mechanisms used to measure the performance of the spam filter. In here, we introduce the performance measures used through the thesis. Let $N=A+B+C+D$ be the total number of test e-mails in our corpus.

	Spam E-mail	Legitimate E-mail
Filter Decision: Spam E-mail	A	B
Filter Decision: Legitimate E-mail	C	D

Table 2.1: Confusion Matrix.

If Table 2.1 denotes the confusion matrix of the e-mail classifier, then we define the accuracy, precision, recall, and F1 for spam e-mails filter as follows:

$$ACCURACY = \frac{A+D}{N},$$

$$PRECISION\ (P) = \frac{A}{A+B},$$

$$RECALL\ (R) = \frac{A}{A+C},$$

$$F1 = \frac{2PR}{P+R}.$$

Similar measures can be defined for legitimate e-mails:

$$ACCURACY = \frac{A+D}{N},$$

$$PRECISION\ (P) = \frac{D}{C+D},$$

$$RECALL\ (R) = \frac{D}{B+D},$$

$$F1 = \frac{2PR}{P+R}.$$

2.4 Testing Corpora

Unlike general text categorization tasks where many standard benchmark collections exist, relatively few spam corpora are available, and the sizes are often small. This is probably because while it is easy to collect spam e-mails, it is much harder to collect legitimate mails for the reason of protecting personal privacy. A common approach to compile spam corpus is to mix legitimate mails from one source with spam messages collected from another place. This may lead to a bias when training the classifier: the corpus compiled may not reflect the true distribution of spam mails and tend to be easily separable. A better way may be collecting both legitimate and spam messages from the same source (for instance, a publicly available mailing list that is heavily spammed). For this reason publicly available spam corpora are used in this work. Among the popular corpus are PU1 [13] and Ling-corpus [13].

PU1 corpus consists of 1099 messages; 481 of which are marked as spam and 618 are labeled as legitimate, with a spam rate of 43.77%. The messages in PU1 corpus have header fields and HTML tags removed, leaving only subject line and mail body text. To address privacy, each token was mapped to a unique integer. The corpus comes in four versions: with or without stemming and with or without stop word removal.

The Ling-spam corpus was collected by the same author of PU1 corpus, which includes: 2412 legitimate messages from a linguistic mailing list and 481 spam messages collected by the author with a 16.63% spam rate. Like PU1 corpus, four versions are available with header fields, HTML tags, and attachments removed.

Since the Ling-spam corpus was compiled from different sources: the legitimate messages came from a spam-free, topic-specific mailing list, and the spam mails were

collected from a personal mailbox, the mail distribution is less like that of the normal user's mail stream, which may make messages in Ling-spam corpus easily separable. In our experiment, we use Lemmatize enabled, stop-list enabled version from PU1 corpus. We chose 62 legitimated e-mails and 48 spam e-mails for testing and the rest 989 e-mails for training.

2.5 Spam Filtering Algorithm

The success of machine learning techniques in text categorization has recently led researchers to explore the applicability of learning algorithms in anti-spam filtering. A supervised learning algorithm is fed with a corpus of messages that have been classified manually as spam or legitimate, and builds a classifier, which is then used to detect incoming spam messages. Apart from collecting separately spam and legitimate training messages, the learning process is fully automatic, and can be repeated to tailor the filter to the incoming messages of particular users or groups, or to capture changes in the characteristics of spam messages.

K-nearest neighbor classification is an instance-based learning algorithm that has shown to be very effective in text classification. The success of this algorithm is due to the availability of effective similarity measure among the *K* nearest neighbor.

The algorithm starts by calculating the similarity between the test e-mail and all e-mail in training set. Then it picks the *K* closet instances and assigns the test e-mail to the most common class among these nearest neighbors.

This after transforming the training e-mails and test e-mails into their TD-IDF vector representation, the next step is to find out the K vectors from training vectors which are most similar to the test vector. In our work, we used the following similarity measure:

$$Sim\ (d_i, d_j) = \sqrt{\sum_{k=1}^{n} (d_{ik} - d_{jk})^2}$$

Here d_i and d_j are two e-mail vectors of dimension n. While the Euclidian distance is a measure for the similarity between vectors in the formula above, other measures are also possible.

Then we calculate the weight of the spam class and legitimate class based on the formula [8]:

$$p(\vec{x}, C_j) = \sum_{\vec{d}_i \in KNN} Sim(\vec{x}, \vec{d}_i) y(\vec{d}_i, C_j)$$

where $\vec{d}_i$ is one of the neighbors in the training set. $\vec{x}$ is the test e-mail. $y(\vec{d}_i, C_j)$ is equal to 1 if $\vec{d}_i$ belongs to C_j, otherwise $y(\vec{d}_i, C_j)$=0.

The final step is to compare the weight and classify the test e-mail.

Because K-nearest neighbor classification is a simple way to apply to our junk e-mail problem, we use this filter algorithm in our experiment. If K is too large, big classes will overwhelm small ones. On the other hand, if K is too small, the advantage of KNN algorithm, which could make use of many experts, will not be exhibited [9]. Throughout our experiments, we set K=30 (this value was chosen because our experiments show that it provides a good compromise between performance and accuracy). Table 2.2 shows the performance of the e-mail filter.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
88.1%	85.7%	87.5%	86.6%	90.1%	88.7%	89.4%

Table 2.2: KNN filter result without feature selection algorithm.

2.6 Conclusion

In this chapter, we presented some basic concepts related to content based e-mail filters. We also introduced the dataset and performance measures used throughout our thesis. The experimental results in Table 2.2 (obtained using a K-nearest neighbor filter classifier without any feature selection algorithm) will be used as a baseline for comparison with other feature selection based systems proposed throughout the thesis.

Chapter 3

Some Traditional Algorithms for Dimensionality Reduction

3.1 Introduction

There are many possible techniques for dimensionality reduction. Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two commonly used techniques for dimensionality reduction. Linear Discriminant Analysis easily handles the case where the within-class frequencies are unequal and their performances have been examined on randomly generated test data. This method maximizes the ratio of between-class variance to the within-class variance in any particular data set, thereby guaranteeing maximal separability. In PCA, the shape and location of the original data sets changes when transformed to a different space whereas LDA doesn't change the location but only tries to provide more class separability and draw a decision region between the given classes. In this chapter, we also present the results obtained using Singular Value Decomposition (SVD).

3.2 Principle Component Analysis (PCA)

Principle Component Analysis (PCA) [16] is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. PCA is also an eigenvector method designed to model linear variability in high dimensional data. In PCA, one computes the linear projections of greatest variance from the top eigenvectors of the data covariance matrix. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. It is a common technique for finding patterns in data of high dimension.

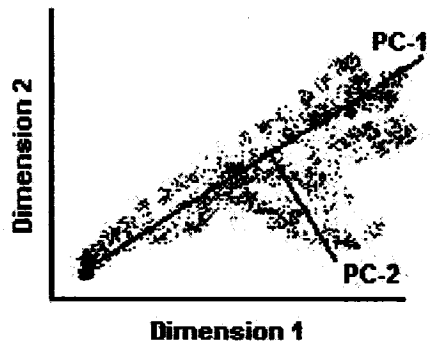

Figure 3.1: Principle Components.

The two lines denoted by PC-1 and PC-2 in Figure 3.1 represent 2 consecutive principle components. Principle Components is a set of variables that define a projection that encapsulates the maximum amount of variation in a dataset and is orthogonal and therefore uncorrelated to the previous principle component of the same dataset.

PCA is a classical tool that has been used successfully for dimensionality reduction in many applications including spam filters. First, we subtract the mean from each of the

data dimensions. Next we calculate the covariance matrix. Then we calculate the eigenvectors and eigenvalues of the covariance matrix. Once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalues, highest to lowest. This gives us the components in order of significance. After this, we can ignore the components of less significance. Although we may lose some information, however, if the eigenvalues are small, we don't lose much. If we leave out some components, the final data set will have fewer dimensions than the original. To be precise, if we originally have N dimensions in our data, and so we calculate N eigenvectors and eigenvalues, and then we choose only the first M eigenvectors, then the final data set has only M dimensions. The value of M was chosen by varying M between 1 to N and choosing the value of N that corresponds to the best accuracy.

Table 3.1 shows the result obtained by our experiments using the features selected with PCA for M=38 (Our experiments showed that this value of M obtained the best classification accuracy). In this case, 5 spam e-mails out of 48 cases and 6 legitimate e-mails out of 62 legitimate cases were misclassified.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
90.0%	87.7%	89.6%	88.7%	91.8%	90.3%	91.1%

Table 3.1: Experimental results for PCA-based features.

3.3 Singular Value Decomposition (SVD)

SVD is an efficient algebraic feature extraction method and it is also a well-known matrix factorization technique that factors a $m \times n$ matrix A into three matrices as follows [18]:

$$A = U\,SV^{T} \qquad (3.1)$$

where U and V are orthogonal matrices, and S is a diagonal matrix. The columns U_i and V_i of U and V are called the left and right singular vectors, respectively. The diagonal elements of S_i are called the singular values.

The SVD has a variety of applications in scientific computing, signal processing, including spam filters. In our experiment, we neglect the small singular values in the matrix S in the SVD, and hence we obtain matrix approximations whose rank equals the number of remaining singular values. Since the singular values appear in decreasing order, the formula for the matrix approximation becomes:

$$A_K = u_1 s_1 v_1^{T} + \ldots\ldots + u_k s_k v_k^{T}$$

where k is the number of retained singular values. Although we may lose some information, however, if the singular values are small, we don't lose much. If we leave out some components, the final data set will have fewer dimensions than the original.

To be precise, if we originally have N dimensions in our data A, and so we decompose A by equation (3.1). Then we choose only the first K singular values, and the final data set has only K dimensions. The SVD results reported in our experiment represent the best results obtained by the SVD by varying the number of selected features K between 1 to N.

Table 3.2 shows the *best* result obtained by our experiments using the features selected with SVD. Five spam e-mails out of 48 cases and 6 legitimate e-mails out of 62 legitimate cases were misclassified based on SVD features.

Performing PCA is the equivalent of performing Singular Value Decomposition (SVD) on the covariance matrix of the data.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
90.0%	87.7%	89.6%	88.7%	91.8%	90.3%	91.1%

Table 3.2: Experimental results for SVD based features.

3.4 Linear Discriminant Analysis (LDA)

LDA is a well-known technique for dealing with the class-separating problem and also can reduce the dimensions of the dataset. LDA can be used to determine the set of the most discriminate projection axes. After projecting all the samples onto these axes, the projected samples will form the maximum between-class scatter and the minimum within-class scatter in the projective feature space [17].

Let $x_1 = \{x_1^1 ....x_{l1}^1\}$ and $x_2 = \{x_1^2 .....x_{l2}^2\}$ be samples from two different classes. The linear discriminant is given by the vector W that maximizes [19]

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

where

$$S_B=(m_1-m_2)(m_1-m_2)^T \qquad S_W=\sum_{i=1,2}\sum_{x\in x_i}(x-m_i)(x-m_i)^T$$

are the between and within class scatter matrices respectively and m_i is the mean of the classes. The intuition behind maximizing $J(w)$ is to find a direction which maximizes the projected class means (the numerator) while minimizing the classes variance in this direction (the denominator). After we find the linear transformation matrix W, the dataset can be transformed by $y = w^T \times x$. For the c-class problem, the natural generalization of linear discriminant involves $c\text{-}1$ discriminant functions. Thus, the projection is from a d-dimensional space to a $(c\text{-}1)$-dimensional space [20]. For our case, we need to classify the data into two classes, thus is projected into a line.

Table 3.3 shows the result obtained by our experiments using the features selected with LDA. Nine spam e-mails out of 48 cases and 2 legitimate e-mails out of 62 legitimate cases were misclassified based on LDA features.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
90.0%	95.1%	81.3%	87.6%	87.0%	96.8%	91.6%

Table 3.3: Experimental results for LDA based features.

3.5 Conclusion

In this chapter, we presented the results obtained using three traditional classifier independent dimensionality reduction algorithms: PCA, LDA and SVD. Our

experimental results show that all the above three algorithms not only reduced the dimensions of e-mail vectors, but also improved the performance of spam filter. Figure 3.2 shows the best accuracy obtained by the approaches proposed in the chapter.

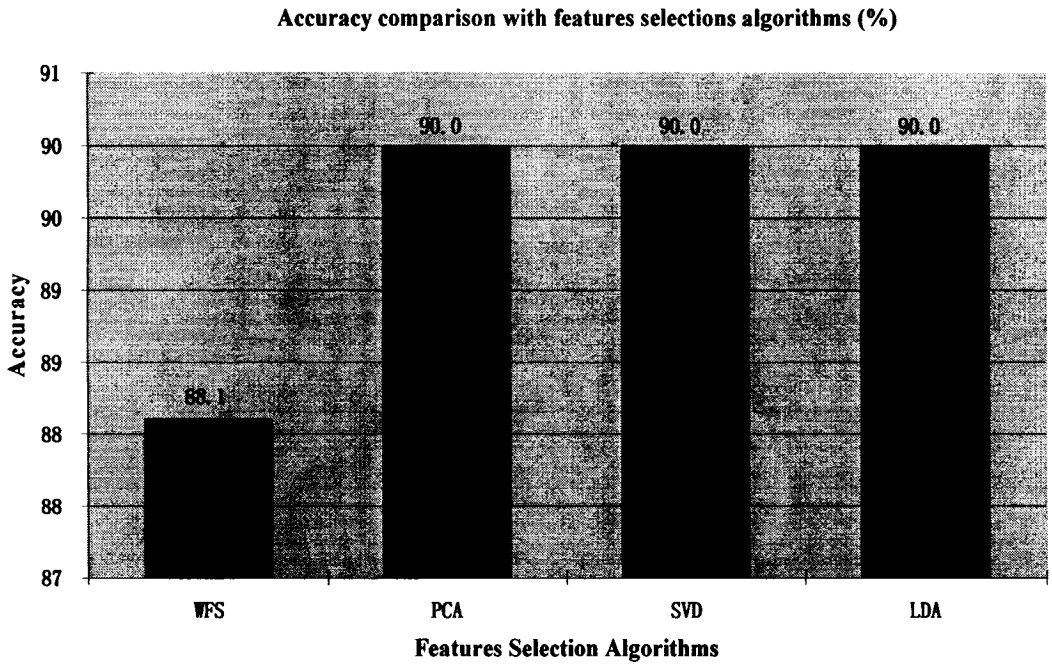

Figure 3.2: Accuracy comparison of traditional algorithms.

Chapter 4

Feature Selection Based On Local Search Algorithms

4.1 Introduction

Local search methods form a very general class of heuristics to treat discrete optimization problems. Such problems are given by a finite set S of feasible solutions and an objective function. The goal is to find a solution with a minimal or maximal objective value [21]. For our e-mail problem, we consider maximization of the accuracy problems of the e-mail filter.

Generally speaking, local search methods move iteratively through the solution set S. Based on the current and maybe on the previous visited solutions, a new solution is chosen. The choice of the new solution is restricted to solutions that are somehow close to the current solution (i.e., in the neighborhood of the current solution).

In this chapter, we present some algorithm based on local search and apply them to our junk e-mail problem. First we introduce the traditional local search algorithm such as Hill Climbing (HC), Simulated Annealing (SA), Tabu Search (TS). We then introduce some techniques inspired by biological systems and artificial life, which can apply to reduce the dimensions of the e-mail. These techniques include Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Artificial Immune Systems (AIS).

Our experiment result shows these techniques can not only reduce the dimensions of the e-mail, but also improve the performance of the junk e-mail filter.

4.2 Hill Climbing (HC)

Hill-Climbing search always moves towards the goal. Using heuristics it finds which direction will take it closest to the goal. The name "hill-climbing" comes from an analogy: A hiker is lost halfway up/down a mountain at night. His camp is at the top of the mountain. Even though it is dark, the hiker knows that every step he takes up the mountain is a step towards his goal. So HC search always goes to the node closest to the goal. The basic idea of HC is to choose a solution from the neighborhood of a given solution, which improves this solution best, and stops if the neighborhood does not contain an improving solution [21].

HC can be used as a features selection for our spam filter problem as follows:

1) Randomly create an initial solution *S1*. This solution corresponds to a binary vector of length equal to the total number of features in the feature set under consideration. The 1's

positions denote the set of features selected by this particular individual. Set $I^*=S1$ and calculate its corresponding accuracy $Y(S1)$.

2) Generate a random neighboring solution $S2$ based on I^* and calculate its corresponding accuracy $Y(S2)$.

3) Compare the two accuracies. If the corresponding accuracy of the neighboring solution $Y(S2)$ is higher than Y(S1), set $I^*=S2$.

4) Repeat step 2 to 3 for a pre-specified number of iteration (or until a certain criterion is reached.)

Although HC has been applied successfully to many optimization problems, it has one main drawback. Since only improving solutions are chosen from the neighborhood, the method stops if the first local optimum with respect to the given neighborhood has been reached. Generally, this solution is not globally optimal and no information is available on how much the quality of this solution differs from the global optimum. Although iterative improvement chooses a solution from the neighborhood of a given solution which improves this solution best and stops if the neighborhood does not contain an improving solution, but only for very special cases does iterative improvement lead to globally optimal solutions. If the underlying neighborhood is exact, i.e., if each local optimum is also a global optimum with respect to the neighborhood [22], then iterative improvement always leads to a global optimum, independent of the initial solution.

A first attempt to overcome the problem of getting stuck in a local optimum was to restart iterative improvement several times using different initial solutions (multiple restart). All the resulting solutions are still only locally optimal, but one can hope that the next local optimum found improves the best found solution so far.

In our experiments, we used 10 different initial solutions and 5000 iteration trying to find the best solution as shown in Table 4.1 and Table 4.2. The average accuracy of the 10 group solutions is 90.9%. The max accuracy of the 10 group solutions is 93.6% and the min accuracy is 86.4%. By different initial solution, we reached the best solution of 93.6% and 4 spam e-mails out of 48 cases and 3 legitimate e-mails out of 62 cases were misclassified. For HC, the initial solution affects the final result very much. The accuracy of group 7 even had a worse performance compared to original system without features selection. Figure 4.1 shows how the accuracy of the spam filter varies with the number of iteration of the HC feature selection algorithm (for the initial state that obtained the best accuracy).

Group	1	2	3	4	5	6	7	8	9	10
Accuracy (%)	90.0	90.0	92.7	92.7	92.7	93.6	86.4	91.8	88.2	90.9

Table 4.1: Accuracy with different initial solutions obtained by HC-based features.

Average	Minimum	Maximum	Variance
90.2%	86.4%	93.6%	0.00053366

Table 4.2: Accuray result of HC.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
93.6%	93.6%	91.7%	92.6%	93.7%	95.2%	94.4%

Table 4.3: Best classification result obtained by HC-based features.

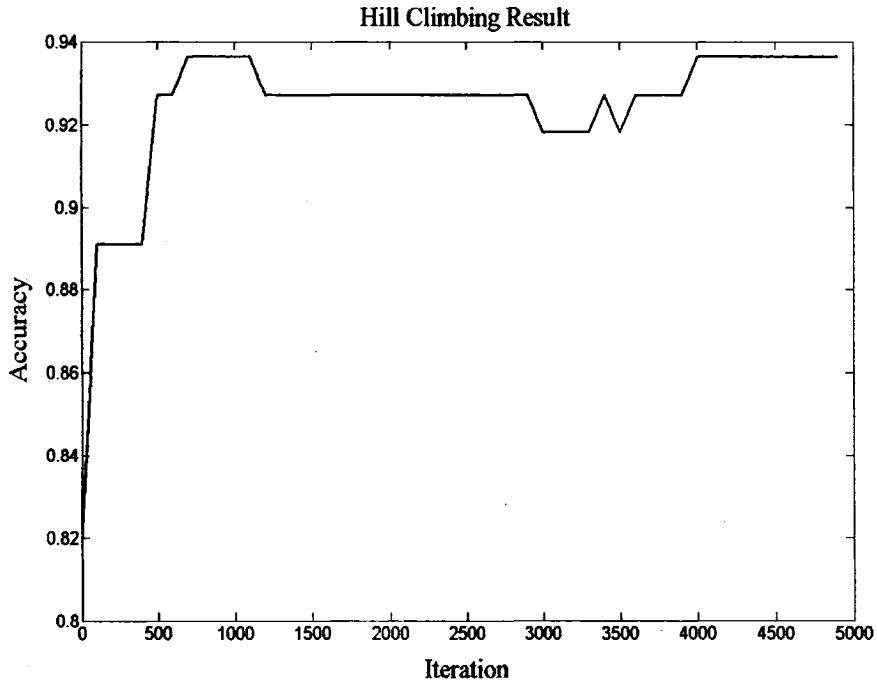

Figure 4.1: Accuracy of HC-based features versus the number of iterations.

4.3 *Simulated Annealing (SA)*

Simulated annealing is a generalization of a Monte Carlo method for examining the equations of state and frozen states of n-body systems [23]. The concept is based on the manner in which liquids freeze or metals recrystalize in the process of annealing. In an annealing process a melt, initially at high temperature and disordered, is slowly cooled so

that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered and approaches a "frozen" ground state at $T=0$. Hence the process can be thought of as an adiabatic approach to the lowest energy state. If the initial temperature of the system is too low or cooling is done insufficiently slowly the system may become quenched forming defects or freezing out in metastable states.

Kirkpatrick *et al.* [24] proposed SA as a local search technique in 1983. It merges HC with the probabilistic acceptance of non-improving moves. Similar to HC, SA iteratively constructs a sequence of solutions where two consecutive solutions are neighbored. However, for SA the next solution does not necessarily have a better objective value than the current solution. This makes it possible to leave local optimum.

First, a solution is chosen from the current solution. Afterwards, depending on the difference between the objective values of the chosen and the current solution, it is decided whether we move to the chosen solution or stay with the current solution. If the chosen solution has a better objective value, we always move to this solution. Otherwise we move to this solution with a probability which depends on the difference between the two objective values. More precisely, if *S1* denotes the current solution and *S2* is the chosen solution, we move to *S2* with probability:

$$P(s_1, s_2) = e^{-\max\{Y(s_1)-Y(s_2),0\}/T},$$

where Y(S) indicates the cost function associated with solution S. Parameter T is a positive control parameter (temperature) which decreases with increasing number of iterations and converges to 0. As the temperature is lowered, it becomes ever more difficult to accept worsening moves. Eventually, only improving moves are allowed and

the process becomes 'frozen'. The algorithm terminates when the stopping criterion is met [25]. Furthermore, the probability above has the property that large deteriorations of the objective function are accepted with lower probability than small deteriorations.

SA based features selection algorithm used in this thesis can be summarized as follows:

1) Randomly create an initial solution *S1*.This solution corresponds to a binary vector of length equal to the total number of features in the feature set under consideration. The 1's positions denote the set of features selected by this particular individual. Set $I^*=S1$ and calculate its corresponding accuracy *Y(S1)*.

2) Create the parameter (temperature) T and constant cooling factor α in the range (0...1).

3) Generate a random neighboring solution *S2* based on I^* and calculate its corresponding accuracy.

4) Compare the two accuracies. If the corresponding accuracy of the neighboring solution *Y (S2)* is higher than *Y(S1)*, set $I^*=S2$. Otherwise, generate $U = rand\ (0...1)$. Compare U and *P(S1,S2)*.If $U > P(S1,S2)$, $I^*=S2$.

5) Decrease the temperature by $T=T\times \alpha$.

6) Repeat step 3 to 5 for a pre-specified number of iteration (or until a certain criterion is reached).

To compare with the HC algorithm, we set the same 10 initial solutions. Four spam e-mails out of 48 spam cases and 1 legitimate e-mail out of 62 legitimate cases were misclassified based on SA features. From table 4.5, the average accuracy reached 92.2% which is better than that of Hill Climbing and the max accuracy is 95.5% which is high than the result of HC and that without FSS. Figure 4.2 shows the accuracy varies with the

number of iteration. This figure is produced for group 3. At each temperature, we set the iteration to 100. In our experiment, the temperature decreased 50 times.

The major difficulty in implementation of the algorithm is that there is no obvious analogy for the temperature T with respect to a free parameter in the combinatorial problem. Furthermore, avoidance of local optimization is dependent on the "annealing schedule", the choice of initial temperature, how much iteration are performed at each temperature, and how much the temperature is decremented at each step as cooling proceeds.

Group	1	2	3	4	5	6	7	8	9	10
Accuracy (%)	88.2	89.9	95.5	95.5	90.0	93.6	93.6	92.7	92.7	90.9

Table 4.4: Accuracy of SA-based features for10 different initial solutions.

Average	Minimum	Maximum	Variance
92.2%	88.2%	95.5%	0.00064630

Table 4.5: Accuray results of SA.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
95.5%	92.2%	97.9%	94.6%	98.3%	93.6%	95.9%

Table 4.6: Best accuracy classification result by SA-based features.

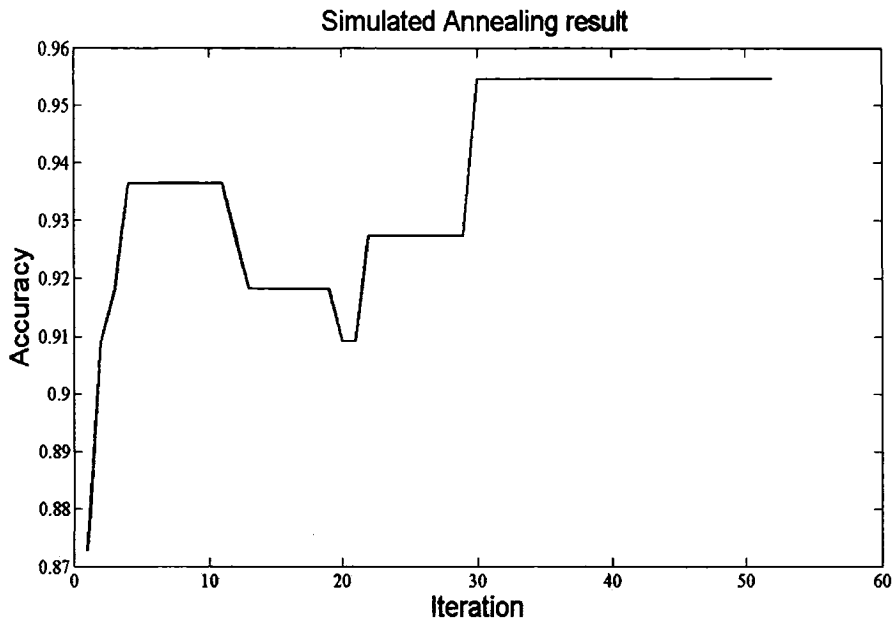

Figure 4.2: Accuracy of SA-based features versus the number of iterations.

Under certain conditions for the neighborhoods and the way in which the control parameter T is decreased, it is possible to prove that SA is asymptotically an optimization algorithm. The proof is based on results on Markov chains. If the neighborhood is connected, i.e. if it is possible to move from each solution via a sequence of neighbored solutions to each other solution, and the control parameter T does not decrease too quickly to 0, the SA algorithm converges to a globally optimal solution with probability 1 [26]. However, this result is mainly of theoretical interest. Each concrete realization is still a heuristic method, since it only runs a finite number of iterations. Furthermore, the above result does not give a rate of convergence. Thus, no bounds on the quality of the solution after a finite number of steps are known [21].

4.4 Threshold Accepting (TA)

A variant of SA is the threshold accepting method. It was designed by Ducek and Scheuer [21] as a partially deterministic version of simulated annealing. The only difference between SA and TA is the mechanism of accepting the neighboring solution. Where SA uses a stochastic model, TA uses a static model: if the difference between the objective value of the chosen and the current solution is smaller than a threshold T, we move to the chosen solution. Otherwise it stays at the current solution [21]. Again, the threshold is a positive control parameter which decreases with increasing number of iterations and converges to 0. Thus, in each iteration, we allow moves which do not deteriorate the current solution more than the current threshold T and finally we only allow improving moves.

TA based features selection algorithm used in this thesis can be summarized as follows:

1) Randomly create an initial solution *S1*.This solution corresponds to a binary vector of length equal to the total number of features in the feature set under consideration. The 1's positions denote the set of features selected by this particular individual. Set $I^*=S1$ and calculate its corresponding accuracy *Y(S1)*.

2) Create the parameter (temperature) T and constant cooling factor α in the range (0 . .1).

3) Generate a random neighboring solution *S2* based on I^* and calculate its corresponding accuracy.

4) Compare the two accuracies *Y(S2)* and *Y(S1)*. If the corresponding accuracy of the neighboring solution *Y(S2)* is higher than *Y(S1)*, set $I^*=S2$. Otherwise, set β = *Y (S2)-Y (S1)*. Compare β and T. If $T > \beta$, $I^*=S2$.

5) Decrease the temperature by $T=T\times\alpha$.

6) Repeat step 3 to 5 for a pre-specified number of iteration (or until a certain criterion is reached).

The main advantage of TA is its conceptual simplicity. TA simplifies SA procedure by leaving out the probabilistic element in accepting worse solutions. Instead it introduces a deterministic threshold, T, and a worse solution is accepted if its difference to the incumbent solution is smaller or equal to the threshold. Compared with SA, step 4 is different for the two algorithms.

To compare with the HC and SA algorithm, we also set the same 10 initial solutions. Three spam e-mails out of 48 spam cases and 3 legitimate e-mails out of 62 legitimate cases were misclassified based on TA features. From table 4.8, the average accuracy reached 92.0% which is better than that of HC and the max accuracy is 94.6% which is high than the result of HC and that without FSS. Figure 4.3 shows the accuracy varies with the number of iteration. This figure is produced for group 3.

Group	1	2	3	4	5	6	7	8	9	10
Accuracy (%)	91.8	92.7	94.6	91.8	92.7	91.8	93.6	90.0	90.9	90.0

Table 4.7: Accuracy of TA-based features for10 different initial solutions.

Average	Minimum	Maximum	Variance
92.0%	90.0%	94.6%	0.00021715

Table 4.8: Accuray results of TA.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
94.6%	93.8%	93.8%	93.8%	95.2%	95.2%	95.2%

Table 4.9: Best accuracy obtained by TA based features.

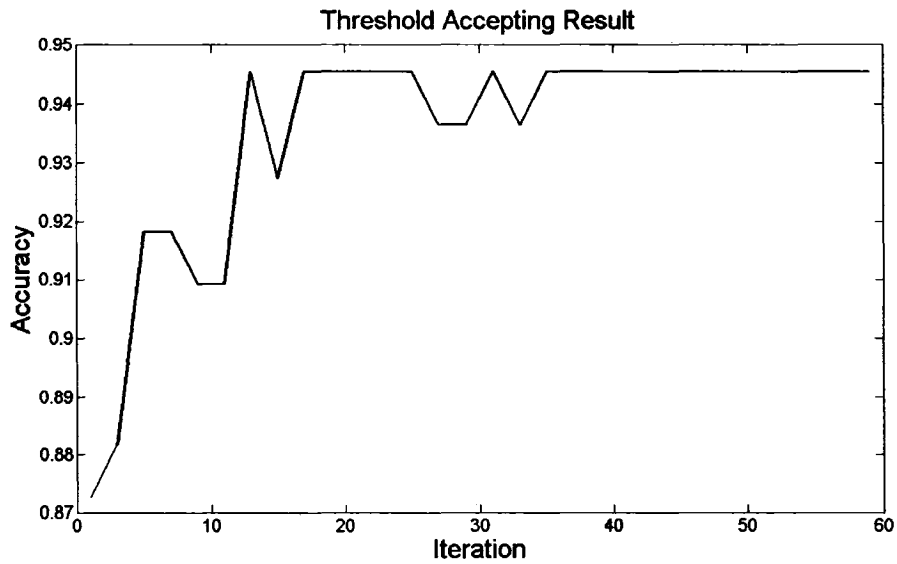

Figure 4.3: Accuracy of TA-based features versus the number of iterations.

4.5 TABU SEARCH (TS)

Tabu (or taboo) comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga island to indicate things that cannot be touched because they are sacred. According to Webster's Dictionary, the word also means "a prohibition imposed by social custom as a protective measure" or of something "banned as constituting a risk." These current more pragmatic senses of the word accord well with the theme of TS.

Glover [27] [28] proposed TS to evaluate better solutions in combinatorial optimization problems. Many of the heuristic search algorithms have the drawback that it can easily get stuck in a local minimum. TS avoids this problem using a very simple, yet elegant, idea. In TS, an adaptive memory, called tabu list, is introduced into the search method. TS continues the solution search process after storing some attributes in the tabu list for a period, making them fixed and escaping from a local minimum. The length of tabu list is called tabu length. The length of the tabu list may be fixed (static tabu list) or may vary (dynamic tabu list) and a solution is declared tabu if it is an element of the tabu list. Because of the finite length of the tabu list, the above realization of the idea to forbid solutions already visited and, thus, to avoid cycling, cannot prevent cycling completely, but at least it makes cycling unlikely.

Because the tabu list may eliminate some promising solutions from the search process, other criteria, called aspiration criteria, are introduced. These criteria may overrule the tabu state of a solution.

In our work, we adapted TS for the e-mail features selection as follows:

1) Randomly create an initial solution I .This solution corresponds to a binary vector of length equal to the total number of features in the feature set under consideration. The 1's positions denote the set of features selected by this particular individual. Set $I^*=I$.

2) Define a first-in first-out (FIFO) tabu list of fixed length N. Push the initial solution I into the Nth position in the tabu list.

3) Mutate the solution I^* with a proper rate to find the neighbor solutions of the initial solution.

4) Sort the neighbor solutions based on their corresponding accuracy then select the solution with the highest value.

5) Check if this best solution is in tabu list. If yes, choose the next. Repeat this selection process until find out a solution which is not in the tabu list. Set I^* equal to this solution.

6) Push I^* found in 5 into the tabu list. If the length of the tabu list exceeds N, then the first solution in the tabu list is dropped from it.

7) Repeat step 3 to 7 for a pre-specified number of iteration (or until a certain criterion is reached.)

When the algorithm terminates, it reports the best individual and its affinity, i.e., the accuracy of the classifier when using only the subset of features corresponding to this individual. It also outputs the obtained accuracy when using the test corpus.

Table 4.10 shows the result obtained by our experiments using the features selected with TS. In this case, we set the tabu list length to 20 and 3 spam e-mails out of 48 cases and 3 legitimate e-mails out of 62 legitimate cases were misclassified based on TS features. Figure 4.4 shows how the accuracy of the spam filter varies with the number of iteration of the TS feature selection algorithm.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
94.6%	93.8%	93.8%	93.8%	95.2%	95.2%	95.2%

Table 4.10: Best accuracy results obtained by TS based features.

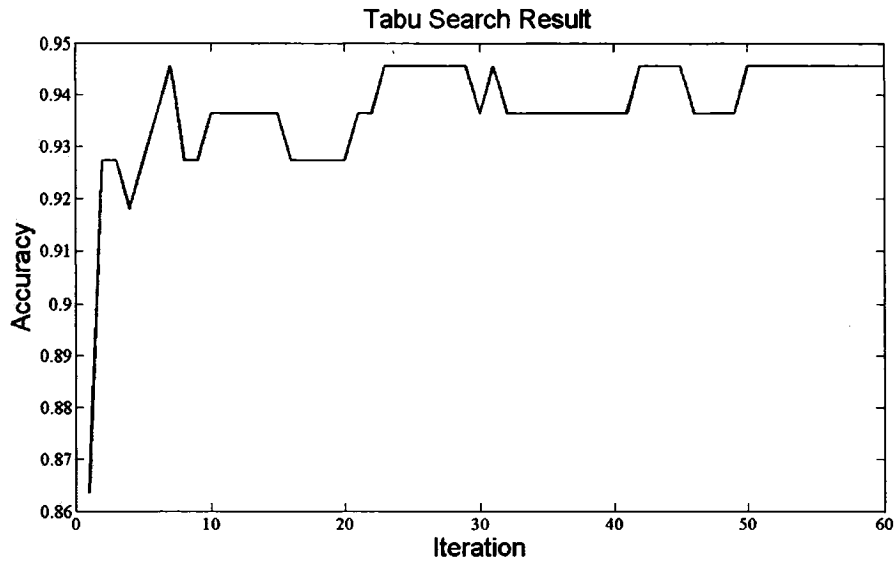

Figure 4.4: Accuracy of TA-based features versus the number of iterations.

4.6 *Particle Swarm Optimization (PSO)*

PSO [29] is a new population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. Consider a group of birds are randomly searching food in an area where there is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So the best strategy to find the food is to follow the bird which is nearest to the food [30].

PSO learned from the scenario above and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

In our experiments, we define the fitness of each particle as the recognition accuracy corresponding to the features selected by this particle using a pre-specified classifier. The algorithm described in here is a slightly modified version of the PSO algorithm to fit the binary nature of the feature selection problem.

Let v_{ij} and p_{ij} denote the j^{th} component of the i^{th} particle velocity v_i and position P_i respectively. The system is initialized with a group of random particles with $0 \le p_{i,j} \le 1$, and then searches for optima by updating generations. In each iteration, each particle position is updated by following two "best" particles. The first one (denoted by Pbest) is the best fitness it has achieved so far. The fitness value is also stored. Another "best" value that is tracked by the particle swarm optimizer is the best value (denoted by Gbest), obtained so far by any particle in the population. After finding the two best values, the particle updates its velocity and positions with following equations:

$$v_{i,j} = v_{i,j} + c_1 r_1 (pbest_{i,j} - p_{i,j}) + c_2 r_2 (gbest_{i,j} - p_{i,j}) \qquad (4.1)$$

$$v_{i,j} = 2\sigma(v_{i,j}) - 1$$

$$p_{i,j} = f_i(p_{i,j} + v_{i,j}) \qquad (4.2)$$

Where $1 \le r_1, r_2 \le 0$ are uniformly distributed random variables, $c_1, c_2 \ge 0$ are learning factors and $\sigma(x)$ is the sigmoid function given by:

$$\sigma(x) = \frac{1}{1 + \exp(-\alpha x)}$$

and $f_i = (T(p_{i,0}), T(p_{i,1}), \ldots T(p_{i,2^n-1}))$ where:

$$T(x) = \begin{cases} 1, & x \ge 0.5, \\ 0, & x < 0.5. \end{cases}$$

The output of the above-adopted algorithm is a vector composed of ones that correspond to the selected features.

Table 4.11 shows the result obtained by our experiments using the features selected with PSO. In this case, 2 spam e-mails out of 48 cases and 7 legitimate e-mails out of 62 legitimate cases were misclassified based on PCA features. Figure 4.5 shows how the accuracy of the spam filter varies with the number of iteration of the PSO feature selection algorithm.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
91.8%	86.8%	95.8%	91.1%	96.5%	88.7%	92.4%

Table 4.11: Best accuracy classification results obtained by PSO based features.

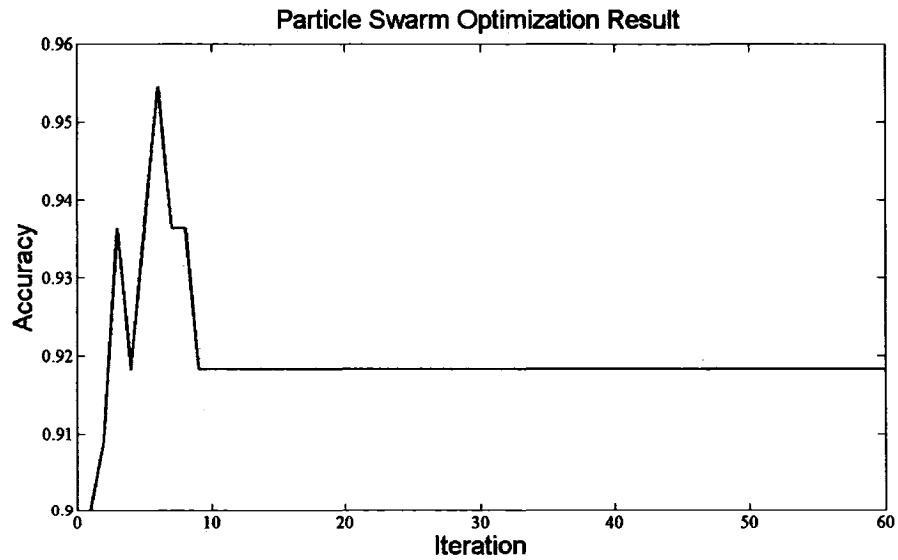

Figure 4.5: Accuracy of PSO-based features versus the number of iterations.

A number of factors will affect the performance of the PSO. Firstly, the number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (less particles) must be sought. In our experiment, we set the number of particles to 300. Learning factor C_1, C_2 are another important factors. C_1 is self confidence factor and C_2 is swarm confidence factor respectively. Usually C_1 C_2 equal to 2.

4.7 Ant Colony Optimization (ACO)

Ant Colony Optimization [31] is another heuristic optimization method for solving optimization problems which borrows ideas from biological ants. In 1999, the Ant Colony Optimization metheuristic was defined by Dorigo, Di Caro and Gambardella [14] [15].

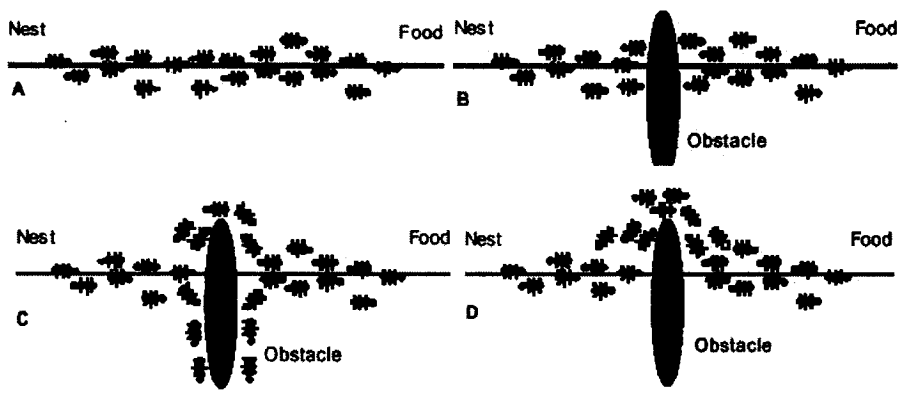

Figure 4.6: Example of real ants. [31]

Consider for example the experimental setting shown in Figure 4.6. There is a path along which ants are walking from food source to the nests (Figure 4.6 (A)). Suddenly an

obstacle appears and the path is cut off see Figure 4.6(B). While the ants walking from food source to the net have to decide whether to turn up or down (Figure 4.6(C)). The choice is influenced by the intensity of the pheromone trails by preceding ants. A higher level of pheromone on the up path gives an ant a stronger stimulus and thus a higher probability to turn up. The first ant reaching the obstacle has the same probability to turn up or down (for there was no previous pheromone on the two alternative paths). Because upper path is shorter than down path, the first ant following it will reach the nest before the first ant following down path (Figure 4.6(C)). The result is that an ant returning from the nest to the obstacle will find a stronger trail on up path, caused by the half of all the ants that by chance decided to approach the obstacle via up path and by the already arrived ones coming via up path: they will therefore prefer up path to down path. Therefore, the number of ants following up path per unit of time will be higher than the number of ants following down path. This causes the quantity of pheromone on the shorter path to grow faster than on the longer one, and therefore the probability with which any single ant chooses the path to follow is quickly biased toward the shorter one. The final result is larger amount of pheromone will accumulate on the shorter paths to good food sources because larger number of ants will cross it back and forth per unit time as compared to longer paths and very quickly all ants will choose the shorter path.

The originally proposed ant colony optimization algorithm fits naturally in optimization problems corresponding to the selection of optimum permutation (such as the traveling sales person problem), i.e., we can apply it for the feature selection problem if we fixed the size of the subset of features to be chosen.

In here, we present a simple algorithm that borrows ideas from the real ant colony but doesn't have the above constraint, i.e., we don't have to pre-determine the size of the optimal feature subset.

The system is initialized with a group of ants moving across a full binary tree of depth n and $2n$ leaves. Each leaf corresponds to one of the possible $2n$ feature subsets. The root of the tree corresponds to the nest of the ants and the accuracy of the classifier based on the feature subset associated with each leaf corresponds to the amount of food found at the food source.

The algorithm proceeds by iterating through the following three basic steps:

1) Construct a solution for all ants: At each node, each ant has to make a (statistical) decision whether to follow the up path or the down path. At the first iteration, all the ants will move randomly. However, on subsequent iterations, the ants' choices will be influenced by the intensity of the pheromone trails left by preceding ants. A higher level of pheromone on the up path gives an ant a stronger stimulus and thus a higher probability to turn up and vice versa. Let $Pher_l\ (U)$ and $Pher_l\ (D)$ denote the value of the pheromone accumulated at the up edge and the down edge of a given node at the l^{th} level of the tree. Then the ants' behavior equivalent to having each ant choosing a uniformly distributed random variable $1 \leq r \leq 0$ and choosing to follow the up edge at the l^{th} level of the tree if, $r \geq \frac{Pher_l(D)}{Pher_l(U) + Pher_l(D)}$ and to follow the down edge otherwise.

2).Do a global pheromone update: For our problem, this step is also different than the one proposed in the original Ant Colony Optimization algorithm. Instead of updating the pheromone along the visited arcs only, we update all the corresponding 2^{l-1} arcs at l^{th} level of the tree. The amount of pheromone laid by each ant corresponds to the amount of

food (i.e., the classifier accuracy) that the ant finds at the leaf of the tree at the end of the path followed by this ant.

3). Evaporate pheromone: after each iteration, a portion of the pheromone of the edge is evaporated according to a local updating rule, such that the probability of the selection of that edge by other ants decreases. This prevents construction of similar paths by the set of ants and increases the diversity of the system. The rate of evaporation provides a compromise between the rate of convergence and reliability of convergence. Fast evaporation causes the search algorithm to be stuck at local optima, while slow evaporation lowers the rate of convergence. After enough iteration of the algorithm, the pheromone of the good edges which are used in constructing of low-cost paths will increase and the pheromone of the other edges will evaporate. Thus, in the higher iterations the probability of constructing low-cost paths increases.

Table 4.12 shows the result obtained by our experiments using the features selected with ACO. In this case, 1 spam e-mails out of 48 cases and 4 legitimate e-mails out of 62 legitimate cases were misclassified based on ACO features. Figure 4.7 shows how the accuracy of the spam filter varies with the number of iteration of the ACO feature selection algorithm.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
95.4%	92.1%	97.9%	94.9%	98.3%	93.5%	95.8%

Table 4.12: Best accuracy classification results obtained by ACO based features.

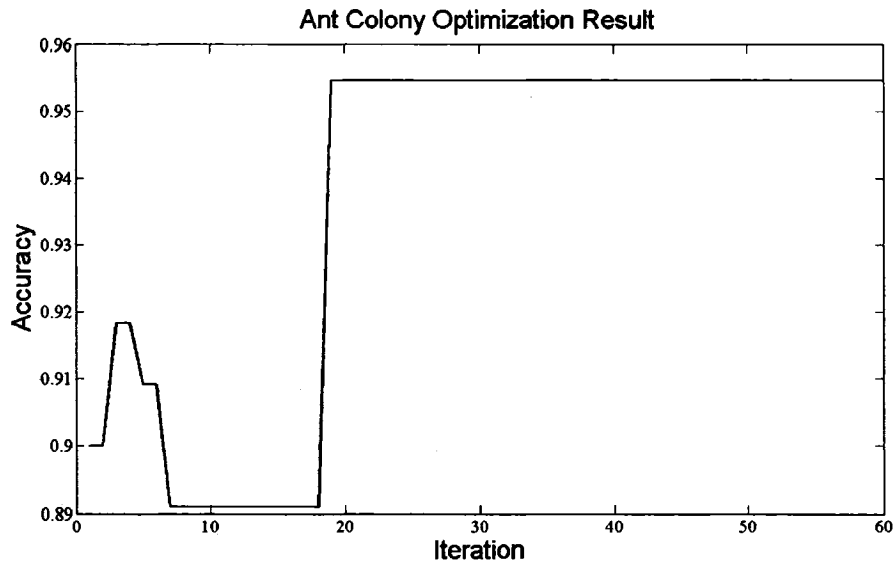

Figure 4.7: Accuracy of ACO-based features versus the number of iterations.

4.8 Genetic Algorithm (GA)

The earliest instances of what might today be called genetic algorithms appeared in the late 1950s and early 1960s, programmed on computers by evolutionary biologists who were explicitly seeking to model aspects of natural evolution. It first was introduced by Holland [32].

GA is based on a biological metaphor: They view learning as a competition among a population of evolving candidate problem solutions. A 'fitness' function evaluates each solution to decide whether it will contribute to the next generation of solutions. Then, through operations analogous to gene transfer in sexual reproduction, the algorithm creates a new population of candidate solutions [33].

GA used in our work can be summarized as follows steps:

1) Randomly initialize a population of individuals (M). Each individual corresponds to a binary vector of length equal to the total number of features in the feature set under

consideration. The 1's positions denote the set of features selected by this particular individual.

2) The affinity of each individual is defined as the recognition accuracy corresponding to the features selected by this individual using a pre-specified classifier.

3) Select the n best highest affinity solution of M. This "Selection" step mimics the principal of "Survival of the Fittest".

4) "Crossover" the n best highest affinity solution. This operation mimics mating in biological populations. The crossover propagates features of good surviving solutions from the current population into the future population, which will have better fitness value on average.

5) "Mutate" these solutions. This operation promotes diversity in population characteristics. The mutation allows for global search and prevents the algorithm from getting trapped in local search

6) Repeat steps 2 to 5 for a pre-specified number of iteration (or until a certain criterion is reached.)

When the algorithm terminates, it reports the best individual and its affinity, i.e., the accuracy of the classifier when using only the subset of features corresponding to this individual. It also outputs the obtained accuracy when using the test corpus.

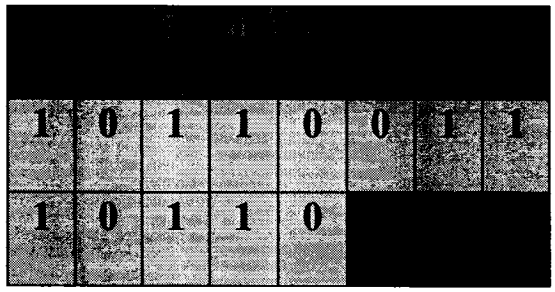

Figure 4.8: Crossover operation of GA.

Figure 4.8 illustrates the effect of crossover genetic operator on individuals in a population of 8-bit strings. It shows two individuals undergoing single-point crossover; the point of exchange is set between the sixth and seventh positions in the genome, producing a new individual that is a hybrid of its progenitors.

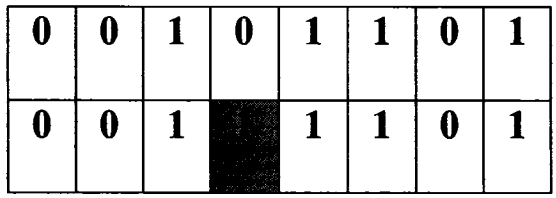

Figure 4.9: Mutation operation of GA.

Fig 4.9 shows an individual undergoing mutation at position 4, changing the 0 at that position in its genome to a 1. A mutation changes the given solution slightly. Mostly, there are several alternatives in changing the solution and the choice for one of them is done randomly. Thus, a mutation is nothing but a random selection of one solution in the neighborhood of the given solution, whereby the neighborhood is defined by the possible changes mutations can perform [21].

Table 4.13 shows the result obtained by our experiments using the features selected with GA. In this case, 5 spam e-mails out of 48 cases and 5 legitimate e-mails out of 62 legitimate cases were misclassified based on GA features. Figure 4.10 shows how the accuracy of the spam filter varies with the number of iteration of the GA feature selection algorithm.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
90.9%	89.6%	89.6%	89.6%	91.9%	91.9%	91.9%

Table 4.13: Best accuracy classification result by GA based features.

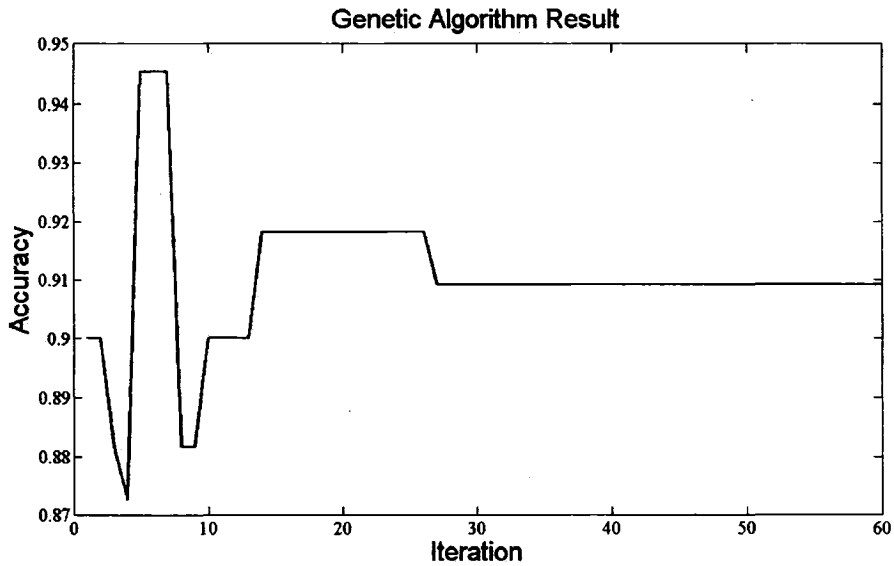

Figure 4.10: Accuracy of GA-based features versus the number of iterations.

Some parameters of GA affect the final result very much. For example, the size of the population, the rate of mutation and crossover must be also chosen with care. If the population size is too small, the genetic algorithm may not explore enough of the solution

space to consistently find good solutions. If the rate of genetic change is too high or the selection scheme is chosen poorly, beneficial schema may be disrupted and the population may enter error catastrophe, changing too fast for selection to ever bring about convergence.

4.9 Artificial Immune Systems (AIS)

Artificial Immune Systems (AIS) [34] are computer algorithms inspired by the principles and processes of the vertebrates' immune system. The algorithms exploit the immune system's characteristics of learning and memory to solve a problem.

The Artificial immune algorithms can be broadly categorized into three subgroups: those using the clonal selection theory, those using negative selection and those using the immune network theory as their main inspiration. In our work, the clonal selection algorithm (CLONALG) [34], a recently proposed optimization technique developed on the basis of clonal selection of the AIS, is adopted and used to select an optimal subset from the constructed features. The set of selected features selected by our algorithm is classifier dependant. This means that different possible feature subsets are examined by the algorithm and the classifier performance is tested for each subset and finally the best discriminatory feature subset is chosen by the algorithm.

The CLONALG used in our work can be summarized as follows steps:

1) Randomly initialize a population of individuals (M). Each individual corresponds to a binary vector of length equal to the total number of features in the feature set under consideration. The 1's positions denote the set of features selected by this particular individual.

2) The affinity of each individual is defined as the recognition accuracy corresponding to the features selected by this individual using a pre-specified classifier.

3) Select the n best highest affinity elements of *M* and generate copies of these individuals proportionally to their affinity.

4) Mutate all these copies with a rate proportional to their affinity with the input pattern: the higher the affinity, the smaller the mutation rate.

5) Add these mutated individuals to the population *M* and reselect m of these maturated individuals to be kept as memories of the systems.

6) Repeat steps 2 to 5 for a pre-specified number of iteration (or until a certain criterion is reached.)

When the algorithm terminates, it reports the best individual and its affinity, i.e., the accuracy of the classifier when using only the subset of features corresponding to this individual. It also outputs the obtained accuracy when using the test corpus.

Table 4.14 shows the result obtained by our experiments using the features selected with AIS. In this case, 4 spam e-mails out of 48 cases and 6 legitimate e-mails out of 62 legitimate cases were misclassified based on AIS features. Figure 4.11 shows how the accuracy of the spam filter varies with the number of iteration of the PCA feature selection algorithm.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
90.9%	88.0%	91.7%	89.8%	93.3%	90.3%	91.8%

Table 4.14: Best accuracy classification result by AIS based features.

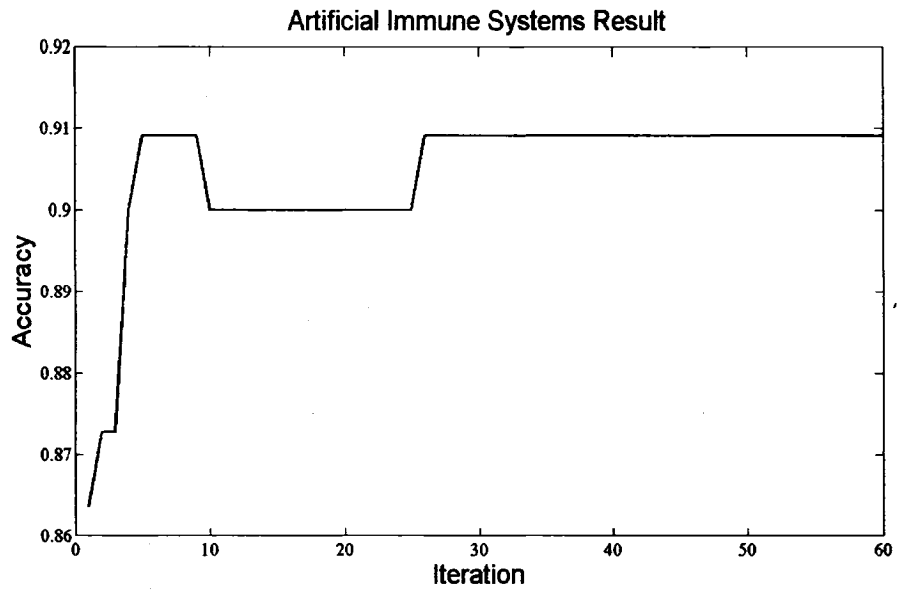

Figure 4.11: Accuracy of AIS-based features versus the number of iterations.

4.10 Conclusion

In this chapter, we applied several local search techniques as feature selection algorithms. Our experimental results show that all these algorithms obtained a better performance compared with the filter performance without any features selections algorithms. Figure 4.12 shows the best accuracy obtained by the traditional local search algorithms proposed in this chapter. Figure 4.13 also shows the best accuracy obtained by the artificial life algorithms in this chapter.

Our experimental results also show that all these local search techniques outperform the three techniques discussed in chapter 3 (i.e., PCA, LDA, and SVD).

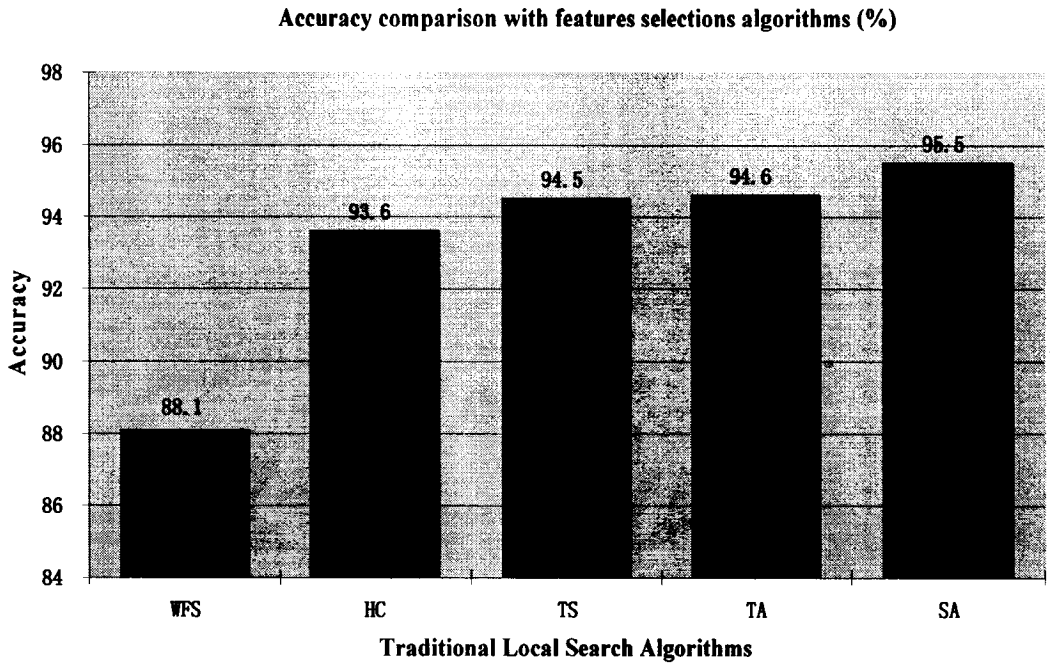

Figure 4.12: Accuracy comparison for traditional local search algorithms.

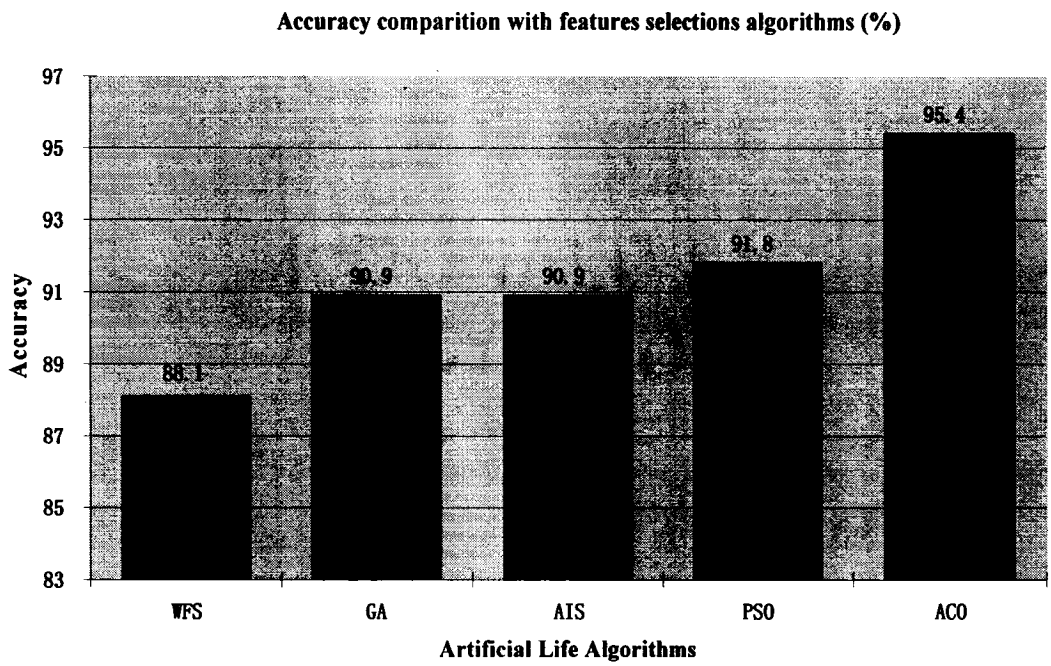

Figure 4.13: Accuracy comparison for artificial life algorithms.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we proposed the use of several optimization techniques as classifier dependent feature selection algorithms for application in spam e-mail filtering. In practice, we should combine several techniques together to solve junk e-mail problem. The performance of our proposed approaches was compared using a simple (KNN) classifier.

Table 5.1 shows the accuracy obtained by the spam filter using non-classifier based dimensionality reduction algorithms; namely PCA, SVD and LDA.

	Without Feature Selection	PCA	SVD	LDA
Accuracy	88.1%	90.0%	90.0%	90.0%
Spam Precision	85.7%	87.7%	87.7%	95.1%
Spam Recall	87.5%	89.6%	89.6%	81.3%
Spam F1	86.6%	88.7%	88.7%	87.6%
Legitimate Precision	90.1%	91.8%	91.8%	87.0%
Legitimate Recall	88.7%	90.3%	90.3%	96.8%
Legitimate F1	89.4%	91.1%	91.1%	91.6%

Table 5.1: Best classification results of traditional algorithms.

Table 5.2 shows the accuracy obtained by the spam filter using different local search feature selection algorithms; namely HC, TA, SA, and TS.

	Without Feature Selection	HC	TA	SA	TS
Accuracy	88.1%	93.6%	94.6%	95.5%	94.5%
Spam Precision	85.7%	93.6%	93.8%	92.2%	93.8%
Spam Recall	87.5%	91.7%	93.8%	97.9%	93.8%
Spam F1	86.6%	92.6%	93.8%	94.6%	93.8%
Legitimate Precision	90.1%	93.7%	95.2%	98.3%	95.2%
Legitimate Recall	88.7%	95.2%	95.2%	93.6%	95.2%
Legitimate F1	89.4%	94.4%	95.2%	95.9%	95.2%

Table 5.2: Best classification results of local search algorithms.

Table 5.3 shows the accuracy obtained by the spam filter using different artificial life techniques, namely ACO, PSO, GA and AIS.

	Without Feature Selection	GA	AIS	PSO	ACO
Accuracy	88.1%	90.9%	90.9%	91.8%	95.4%
Spam Precision	85.7%	89.6%	88.0%	86.8%	92.1%
Spam Recall	87.5%	89.6%	91.7%	95.8%	97.9%
Spam F1	86.6%	89.6%	89.8%	91.1%	94.9%
Legitimate Precision	90.1%	91.9%	93.3%	96.5%	98.3%
Legitimate Recall	88.7%	91.9%	90.3%	88.7%	93.5%
Legitimate F1	89.4%	91.9%	91.8%	92.4%	95.8%

Table 5.3: Best classification results of artificial life algorithms.

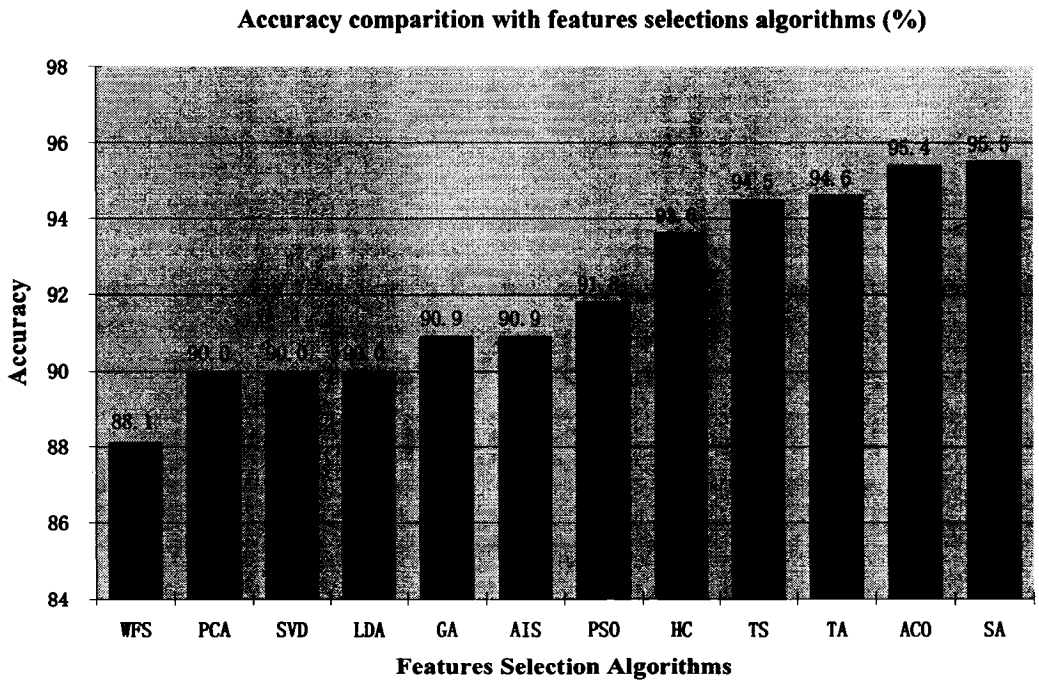

Figure 5.1: Accuracy comparison for all features selection algorithms.

Figure 5.1 compare the accuracy obtained by all the algorithms in this thesis. In Figure 6.1, WFS denotes the system without any feature selection step. From Figure 5.1, it is clear that the accuracy ordering is as follows:

SA>ACO>TA>TS>HC>PSO>GA=AIS>PCA=SVD=LDA.

One should note that the above accuracy ordering should be interpreted with care since it may vary depending on the dataset. However, the conclusion that feature selection strategy improves the spam filter performance should remain unchanged.

5.2 Future Works

Throughout our experiment, we used a simple KNN classifier. We believe that better accuracy can be obtained by the use of a more powerful classifier such as support vector machines. In fact, even the use of Naive Bayes algorithm may dramatically improve the performance of the spam filter (see appendix I).

Since a typical e-mail user may tolerate some spam e-mails, but is not willing to loose any legitimate e-mails, the work in this paper can also be extended by assigning a different weight to different classification errors, i.e., assigning a high penalty to legitimate e-mails classified as spam e-mails.

Other promising techniques such as non negative matrix factorization can also be used. Appendix II shows some preliminary results obtained using LLE.

In order to improve the statistical significance of our results, one may examine the proposed approaches on a larger (unbalanced) database.

Another important problem that needs to be addressed is how to overcome the spammers' trick of modifying the spam terms. For example, several spammers may easily confuse traditional spam filters by using the word "v1agra" instead of "Viagra".

One main disadvantage of these local search techniques (as compared to PCA, LDA, and SVD) is the sensitivity of these techniques to the initial solution and the number of iterations. Further research is required to determine some more guidelines regarding the optimum number of required search iterations.

Appendix I

Naive Bayes Algorithm

Naive Bayes (NB) classifier is a probability-based approach. The basic concept of it is to find whether one e-mail is spam or not by looking at which words are found in the message and which words are absent from it.

Given a feature vector $\vec{d}_j$ = {W_{j1}, W_{j2} . . . $W_{j|\vec{d}_j|}$} of a message, where W_{ji} is the weight of feature *i*, and let c denotes the category to be predicted ($c \in$ *(spam, legitimate)*), by Bayes law the probability that $\vec{d}_j$ belongs to c is:

$$P(c \mid \vec{d}_j) = \frac{P(c).P(\vec{d}_j \mid c)}{P(\vec{d}_j)}$$

Where $P(\vec{d}_j)$ is the probability that a randomly picked document has vector $\vec{d}_j$ as its representation, $P(c)$ is the prior probability of a randomly picked document with label *c*, and $P(\vec{d}_j \mid c)$ denotes the probability of a random picked document with label *c* has $\vec{d}_j$ as its representation.

The denominator $P(\vec{d}_j)$ is the same for all categories under consideration and can be dropped safely. The parameter $P(c)$ can be estimated from training data through

relative frequency. However, it is usually infeasible to compute the term $P(\vec{d}_j \mid c)$ directly. Since the number of possible vectors $\vec{d}_j$ is too high. In order to alleviate this problem, it is common to make the assumption that any two coordinates of the document vector are, when viewed as random variables, statistically independent of each other. So $P(\vec{d}_j \mid c)$ can be decomposed to:

$$P(\vec{d}_j \mid c) = \prod_{i=1}^{|v|} P(W_{ji} \mid c)$$

This is where the name "naive" comes from. Despite the fact that this kind of assumption is often violated in real-world data, naive Bayes classifier performs fairly well on spam filtering task [35] [36] and is already applied in practice.

Table I.1 shows the results obtained by Naive Bayes spam e-mail filter without any features selection algorithms. The accuracy is very high and only 2 legitimate e-mails out of 62 cases were misclassified.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
98.2%	96.0%	100%	97.9%	100%	96.8%	98.4 %

Table I.1: Accuracy of the Naive Bayes filter.

Appendix II

LLE-based Feature Selection for Spam E-mail Filters

In this appendix, we present some preliminary results obtained using the local linear embedding algorithm when applied to the spam filter feature selection problem.

LLE [37] is an unsupervised learning algorithm that computes low dimensional, neighborhood preserving embeddings of high dimensional data. LLE attempts to discover nonlinear structure in high dimensional data by exploiting the local symmetries of linear reconstructions. LLE maps its inputs into a single global coordinate system of lower dimensionality, and its optimizations do not involve local minima. The LLE algorithm is based on simple geometric intuitions and can be described as follows:

Suppose the data consist of N real-valued vectors each of dimensionality D. It is expected that each data point and its neighbors lie on or close to a locally linear patch. The local geometry of these patches is characterized by linear coefficients that reconstruct each data point from its neighbors. In the simplest formulation of LLE, one

identifies nearest neighbors per data point, as measured by Euclidean distance. Reconstruction errors are then measured by the cost function:

$$\varepsilon(W)=\sum_i\left|X_i-\sum_j W_{ij}X_j\right|^2 \qquad \text{(A1)}$$

This adds up the squared distances between all the data points and their reconstructions. The weights summarize the contribution of the j^{th} data point to i^{th} reconstruction. To compute the weights, the cost function should be minimized subject to two constraints: first, that each data point is reconstructed only from its neighbors; second, that the rows of the weight matrix sum to one: The optimal weights subject to these constraints are found by solving a least squares problem. In the final step of the algorithm, each high dimensional observation X is mapped to a low dimensional vector Y representing global internal coordinates on the manifold. This is done by choosing d-dimensional coordinates to minimize the embedding cost function:

$$\phi(Y)=\sum_i\left|Y_i-\sum_j W_{ij}Y_j\right|^2 \qquad (A.2)$$

This cost function is based on locally linear reconstruction errors, but the weights are fixed while optimizing the coordinates Y. The LLE algorithm is simple to implement and is summarized as follows [38]:

1) Compute the neighbors of each data point X_i

2) Compute the weights W_{ij} that best reconstruct each data point X_i from its neighbors minimizing the cost function in (A1).

3) Compute the vectors Y_i by the weight W_{ij} minimizing the quadratic form in (A2).

Table II.1 shows the result obtained by our experiments using the features selected with LLE. In this case, five spam e-mails out of 48 cases and 5 legitimate e-mails out of 62 legitimate cases were misclassified based on LLE features.

Accuracy	Spam Precision	Spam Recall	Spam F1	Legitimate Precision	Legitimate Recall	Legitimate F1
90.9%	89.6%	89.6%	89.6%	91.9%	91.9%	91.9%

Table II.1: Experimental result obtained using LLE based features.

It should be noted that there are some practical problems associated with LLE. The typical application of LLE is for data visualization in which the data is mapped to either 2 or 3 dimensional spaces. However, when applied as a part of a classification system, there is no clear way to determine what dimension to project to. There is also no clear way to determine the number of neighbors. In this work, these two parameters were varied and we chose the values that optimized the classification performance. Also, the LLE doesn't provide an explicit mapping between the original space and the reduced dimensional space. However, since the LLE is unsupervised, in this work, we decided to follow a very simple approach by performing the embedding process for every additional test case. Since the whole process is computationally inexpensive, this doesn't provide any practical problem in our proposed system and the whole embedding process can be performed in few seconds using any reasonable personal computer. However, this simple approach may not be practical for very large sets of data.

References

[1] B. Templeton, "*Reflections on the 25th Anniversary of Spam,*" http://www.templetons.com/brad/spam/spam25.html.

[2] L. Barry and B. Nathaniel, "*A Multifaceted Approach to Spam Reduction,*" First Conference on E-mail and Anti-Spam (CEAS) 2004.

[3] L. Kang, P. Calton and A. Mustaqu, "*Resisting SPAM Delivery by TCP Dampin,*" First Conference on E-mail and Anti-Spam (CEAS) 2004.

[4] S. Cowley, "Microsoft, *NY AG team on lawsuit against spammer,*" InfoWorld, December, 2003.

[5] G. Gross, "AOL, *Earthlink sue alleged spammers,*" InfoWorld, February, 2004.

[6] G. Gross, "Major *ISPs sue hundreds of spammers,*" InfoWorld, March, 2004.

[7] F. Sebastiani, "*Machine learning in automated text categorization,*" ACM Computing Surveys 34, 1, 1–47. 2002.

[8] J. F. Pang, D. B. Bo and S.Bai, "*Research and Implementation of Text Categorization System Based on VSM,*" International Conf. on Multilingual Information Processing [C], pp. 31-36, .2000.

[9] B. Li, S.Yu and Q. Lu, *"An Improved k-Nearest Neighbor Algorithm for Text Categorization,"* 20th International Conference on Computer Processing of Oriental Languages, Shenyang, China, 2003.

[10] N. Soonthornphisaj, K. Chaikulseriwat and P. Tng-on, "*Anti-Spam Filtering: A Centroid-Based Classification Approach,*" 6th IEEE International Conference on Signal Processing, pp. 1096 - 1099, 2002.

[11] L. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos. "*Learning to filter spam e-mail: A comparison of a naive Bayesian and a memorybased approach,*" The Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), 2000.

[12] L. Zhang, J. Zhu and T. Yao, "*An Evaluation Of statistical Spam Filtering Techniques*," ACM Transactions on Asian Language Information Processing, Vol3, NO.4, pp. 243-269, 2004.

[13] http://iit.demokritos.gr/skel/i-config/downloads.

[14] M. Dorigo and G. Di Caro, "*The Ant Colony Optimization Meta-Heuristic,*" In D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization,* McGraw-Hill, 11-32. 1999.

[15] M. Dorigo, G. Di Caro and L. M. Gambardella. "*Ant Algorithms for Discrete Optimization,*" Artificial Life, 5(2):137-172.1999.

[16] R. Gnanadesikan, *"Methods for Statistical Data Analysis of Multivariate Observations,"* second Edition. Wiley, 1997.

[17] L.F. Chen, H.Y.M. Liao, M.T. Ko, J.C. Lin and G.J Yu, *"A new LDA-based face recognition system which can solve the small sample size problem,"* Patten recognition. vol. 33. pp. 1713-1726, 2000.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "*Application of dimensionality reduction in recommender system-a case study,*" ACM WebKDD Web Mining for E-commerce Workshop, 2000.

[19] S. Mika, G. Ratsch, J. Weston, B. Scholkopf and K. Mullers, "*Fisher discriminant analysis with kernels,*" Neural Networks for Signal Processing IX, 1999. pp.41 - 48 Madison, WI Aug. 1999.

[20] R. Duda, P. Hart and D. Stork, "*Pattern Classification,*" John Wiley and Sons, 2001.

[21] J. Hurink, "*Introduction to Local Search,*" http://wwwhome.math.utwente.nl/~hurinkjl.

[22] C. H. Papadimitriou and K. Steiglitz, "*Combinatorial optimization: Algorithms and complexity,*" Prentice Hall, New Jersey. 1982.

[23] N. Metropolis, A. Rosenbluth, and M. Rosenbluth, "*Equation of State Calculations by Fast Computing Machines*", J. Chem. Phys.,21, 6, 1087-1092, 1953.

[24] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "*Optimization by Simulated Annealing,*" Science, pp 671-580, May 1983.

[25] J. A. Clark, J. L. Jacob and S. Stepney, "*The Design of S-Boxes by Simulated Annealing,*" Evolutionary Computation, pp. 1533 – 1537, Vol.2, June 2004.

[26] P.J.M. van Laarhoven and E.H.L. Aarts, "*Simulated annealing: theory and applications,*" D. Reidel Publishing Company, Dordrecht, Holland, 1987.

[27] F. Glover, "*Tabu Search,*" ORSA Journal on Computing, Vol. 1, No. 3, pp. 190-206, 1989.

[28] F. Glover and M. Laguna, *"Tabu search, in Modern Heuristic Techniques for Combinatorial Problems,"* C. Reeves (ed.), Blackwell, Oxford, UK: 70-141, 1993.

[29] R.C. Eberhart and Y.Shi, *"Comparison between genetic algorithms and particle swarm optimization,"* IEEE international conference on evolutionary Comp., pp 611-616, 1998.

[30] http://www.swarmintelligence.org/tutorials.php

[31] M. Dorigo, V. Maniezzo, and A. Colorni, *"The ant system: optimization by a colony of cooperating agents,"* IEEE Transactions on Systems, Man, and Cybernetics-Part B 26(1), pp. 29–41. 1996.

[32] J.H. Holland, *"Adaptation in natural and artificial systems"*, University of Michigan Press, Ann Arbor MI.1975.

[33] G. Luger, *"Artificial Intelligence, Structures and Strategies for Complex Problem Solving,"* fourth Edition, Harlow, England, Addison-Wesley, 2002.

[34] L. N. de Castro and F. J. Von Zuben, *"Learning and optimization using the clonal selection principle,"* Evolutionary Computation, IEEE Transactions on Volume 6, Issue 3, pp.239-251 June 2002.

[35] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras and C. Spyropoulos, *"An evaluation of naive Bayesian anti-spam filtering,"* Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000), 2000.

[36] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, *"A Bayesian approach to filtering junk e-mail,"* In Learning for Text Categorization: Papers from the 1998 Workshop, AAAI Technical Report WS-98-05, Madison, WI.1998.

[37] S. Oweis and L. Saul, *"Nonlinear dimensionality reduction by locally linear embedding,"* Science, vol. 290 no. 5500, pp.2323-2326, Dec. 2000.

[38] S. S. Mohamed, A. M. Youssef , E. F. EL-Sadaany and M. M. A. Salama, *"LLE Based TRUS Image Features Dimensionality Reduction for Prostate Cancer Diagnosis,"* GVIP 05 Conference, CICC, Cairo, Egypt ,19-21 December 2005.

[39] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos and P. Stamatopoulos, *"A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists,"* Information Retrieval, 6, pp. 49–73, 2003.

[40] C. Cortes and V. Vapnik," *Support-vector networks,"* Machine Learning 20, 3, 273–297, 1995.

[41] V. Vapnik, *"The Nature of Statistical Learning Theory,"* Springer-Verlag, New York, 1995.

[42] Abhimanyu Lad, *"SpamNet – Spam Detection Using PCA and Neural Networks,"* 7th International Conference on Information Technology (CIT 2004), pp.205-213, LNCS 3356, Springer Verlag, 2004.