School of Computer Science

A Client-based Anti-spam Filter

Final Year Project

Author: Nazerke Rakhymbayeva
Supervisor: Dr. Ning Zhang

April, 2013
Dedication

This report dedicated to my Mom, Mrs. Rakhymbayeva Gulbanu, who inspired me and gave energy to overcome challenges with enthusiasm, and to the memory of my Dad, Mr. Rakhymbayev Imanbakhyt, who passed respect for education. There is no doubt, that without their love and support this report would not have been made possible.
Abstract

In the age of information technology, computers become an important part. 40 years passed since first electronic mail (email) sent [1]. Nowadays sending email became a daily habit as it is the cheapest and fastest way of communication. Use of Internet services makes life easier, but there are some problems that people face today. According to the information that taken from BBC News, one of the most global Internet problems is receiving spam messages [2]. Programmers generate many software applications to protect mailboxes from spam. However, none of the anti-spam programs is an effective and perfect solution. They are still looking for new techniques with bigger accuracy to be developed in order to catch spam messages.

This report demonstrates how a new client-based anti-spam filter program developed. Detailed information about work during design and implementation period such as building Naive Bayesian classifier, listing algorithms and creating configurable user interface using Java GUI tool provided. Difficulties and justification of decisions in order to build efficient filter throughout the project development described. The benefits of using Java language outlined in the report. Main achievements highlighted by analysing the working program through testing and evaluation period. The program developed successfully as it met all the functional and non-functional requirements stated at the beginning of the development. The final results of the whole project provide a user with two filters, first filter technique based on Naive Bayesian classifier and the second filter technique constructed by mixing functionality of greylisting, blacklisting and whitelisting technologies.

Author: Nazerke Rakhymbayeva
Degree Course: BSc (Hons) Computer Science
Title: A client-based anti-spam filter
Project Supervisor: Dr. Ning Zhang
Date: April 2013
Acknowledgements

I want to say my appreciation and thanks to my supervisor Dr. Ning Zhang for support and encourage throughout the project period. Her suggestion and feedbacks for the provided work have been very useful in completing the entire project. I wish to thanks friends for their contribution during testing and evaluation process. I would like to express sincere gratitude to my family for helping to overcome difficulties during my university life.
Table of Contents

Chapter 1: Introduction ........................................................................................................... 7
  1.1 Motivation......................................................................................................................... 7
  1.2 Structure.......................................................................................................................... 8

Chapter 2: Background and literature survey ......................................................................... 9
  2.1 Spam and Spam categories .......................................................................................... 9
  2.2 The impact of spam ........................................................................................................ 10
  2.3 Spam in numbers............................................................................................................. 11
  2.4 Spam filtering techniques ............................................................................................ 12
  2.5 Existing anti-spam programs ....................................................................................... 14

Chapter 3: Design ................................................................................................................... 16
  3.1 Software design.............................................................................................................. 16
  3.2 System architecture ....................................................................................................... 19
  3.3 Implementation technologies and APIs ....................................................................... 20

Chapter 4: Implementation ................................................................................................... 21
  4.1 A filter based on Naive Bayesian approach .................................................................. 22
  4.2 A filter based on listing methodologies ....................................................................... 26
  4.3 Improvements ............................................................................................................... 29
  4.4 UI Design ...................................................................................................................... 31

Chapter 5: Result .................................................................................................................... 32

Chapter 6: Testing and evaluation ........................................................................................ 36
  6.1 Testing ............................................................................................................................ 36
    6.1.1 Test case 1 ............................................................................................................... 37
    6.1.2 Test case 2 ............................................................................................................... 39
    6.1.3 Test case 3 ............................................................................................................... 39
  6.2 Evaluation ...................................................................................................................... 40

Chapter 7: Conclusions .......................................................................................................... 41

References ............................................................................................................................. 43

Appendix ............................................................................................................................... 46
List of figures and tables

Figure 1: The UML activity diagram of the system ................................................................. 17
Figure 2: The use case diagram of the system ........................................................................ 18
Figure 3: The UML class diagram of the system ................................................................... 18
Figure 4: The structure of the system ..................................................................................... 19
Figure 5: The workflow of the system ................................................................................... 21
Figure 6: The pseudo-code of counting appearance of word .................................................. 22
Figure 7: The pseudo-code of training the dataset method .................................................... 23
Figure 8: Implementing Bayes’ theorem to calculate final probability ..................................... 24
Figure 9: A pseudo-code of combined listing algorithm ......................................................... 26
Figure 10: Part of the actual code with Pattern and Matcher .................................................... 28
Figure 11: The method to list all the files from a folder ......................................................... 29
Figure 12: Old interface ........................................................................................................... 30
Figure 13: Current interface .................................................................................................... 30
Figure 14: A client-based anti-spam filter interface ............................................................... 31
Figure 15: “Open” dialog window .......................................................................................... 33
Figure 16: Checking one message at a time ......................................................................... 34
Figure 17: Checking a set of messages at a time ................................................................... 34
Figure 18: Result part of the application ............................................................................. 35
Figure 19: Result part of the application ............................................................................. 39

Table 1: Workflow of the application ..................................................................................... 32
Table 2: Behaviour of logical operator AND ........................................................................ 35
Table 3: Confusion matrix ....................................................................................................... 36
Table 4: Result of Naive Bayesian classifier .......................................................................... 37
Table 5: Result of Listing methodologies .............................................................................. 38
Table 6: Testing messages using combined filters ................................................................. 38
Table 7: Testing spam and non-spam messages using combined filters .............................. 39
Chapter 1: Introduction

This chapter represents the importance of the project, its purpose and objectives and outlines the content of the report.

1.1 Motivation

Nowadays people live in the age of information technology, where computers are an important part of their lives, and the Internet is necessary for work, study or just to spend time for leisure [3]. Email is the efficient and cheapest way of communication. The rapid development of information and communication technology brings not only advantages, but also certain problems, for example spam. Spam has become a nightmare for Internet users. It clutters mailboxes, filling up disk spaces and costing money with dial-up connection. Users spend a significant amount of time to delete spam messages. Moreover, spam messages use bandwidth and swallow an enormous amount of available spaces in a database, making the Internet busier and slower. Spam sending technology has dramatically improved over the past few years. So did anti-spam software programs. Developers of the anti-spam programs save millions of dollars to the companies, and receive thanks from users who have weaknesses for online shopping, gambling, etc. However, none of the developed anti-spam programs are a perfect solution to the spam problem. Each of them has its own strengths and weaknesses.

In order to start building the entire project, all information related to spam messages and their influences to the people and techniques to overcome problems that spam caused was found. Appropriate information collected by reading and analysing scientific papers and books. For example, article about three most popular spam sending tools and their improvements was written by Henry Stern in his article “A Survey of Modern Spam Tools” was taking about the three most popular spam sending tools and their improvements. He introduced the technique that spammers use to bypass content filtering method [4]. Professors of the Georgia Institute of Technology composed an article about the influence of denial-of-information attack. They underlined two different targets of spam to attack: humans and machines [5]. Another research-paper that was written by a PhD student of the University of Cambridge identifies the actual behaviour of spam attacks in the UK [6]. One of the practical papers was produced by the staff of the University of Michigan and Microsoft Research Silicon Valley. They presented information about triangular spamming, which is the newest technique spammers use to avoid filters [7]. An ambition to create unique anti-spam software program for a final year project has increased due to information gained from research papers.
1.2 Structure

This report represents the process of generating a client-based anti-spam filter program which answers the question like: “How to limit spam without killing an actual electronic message?” At the beginning of the development process, aim and objectives of the project were revealed. The aim of the project was to create a filter with a lower false positive rate. As the title states the program should be client-based filter, which induces to create user-friendly and configurable user interface. The main requirements identified in order to develop powerful software program. First of all the program should give less of accidentally blocking legitimate messages (minimum number of false positive rate) as the cost of it is much higher than letting spam pass the filter. The following requirement states that the program should focus on incoming messages and employ different anti-spam techniques. Using different techniques at a time increases the accuracy of identifying spam messages. Generated client-based anti-spam filter uses two filtering methods. First one based on improved Naive Bayesian classifier that works on the keyword context. The keyword context was created by mapping an existing keyword (word that need to calculate the probability of being spam) to set of related words. The second filter based on greylisting, blacklisting and whitelisting methodologies. The purpose of mixing such techniques was to provide accurate results to sort message to be spam or non-spam.

The report organized as follows: Chapter two gives essential background information that used to create the project, such as the definition of spam, different kinds of spam, techniques to avoid spam messages and tools (software programs) that use given techniques. Determining spam categories improve the efficiency of spam filtering techniques especially of the context filter. Also, the second chapter provides with statistics of the last three years, which helps trace the actual accuracy of existing anti-spam programs. The next chapter describes the actual work carried out during building the project period, including the design of the software and explanation of implementation technology choices. The structure of the actual program pointed out in chapter three. The fourth chapter represents the implementation details such as the process of building algorithms that have been used to create the filter. Also, it provides information about designing the user interface. Chapter five is provided to show the results that emphasize performance of existing project and interaction with users in practice. Screen shots of the executable program are also provided to explain the program in real time running. Chapter six describes detailed knowledge about how the system was tested and evaluated. All the data obtained from different test cases are presented using tables and diagrams. Analysis of the results is demonstrated in this chapter. The final chapter of the report summarises all the works that have been achieved for the project and difficulties that have been met during construction. Learning outcomes and other possible improvements are also highlighted in chapter seven.
Chapter 2: Background and literature survey

This chapter is aimed at providing readers with a suitable basic sense of the topic discussed in the report. Readers gain knowledge about spam and damage caused by spam, after reading the chapter. Also the tools and techniques to overcome spam are indicated. Some statistics of the last few years are demonstrated to readers in order to assure them of the importance of building a strong anti-spam tool.

2.1 Spam and Spam categories

Originally, the word spam appeared as a trademark of canned meat (SPicy hAM) of Hormel Foods in 1937 [8]. The company was trying to sell with an aggressive advertising campaign. Later, all the messages that have sent massively by analogy began to be known as spam. According to the definition provided in a book “Ending spam”, spam can be described as mass mailing commercial, political and other advertising information or other types of messages to the person that does not want to get them [9].

Depending on the aim and objectives of the sender, spam messages may or may not include commercial information. That is why two groups of messages, such as unsolicited commercial emails and unsolicited bulk emails declared. However, there are big amount of variety of spam. The most common spam categories introduced below.

**Advertising spam**: The advertising message always praises real goods or services and provides links to original sites, where users can get more information about them. Site promotion spam advertises remarkably good and/or free products by providing a link to the site that does not have any reasonable information. Nevertheless, the site’s ranking will improve by deceived users. Sometimes, the message can be empty, and the page with the counter program opens in a new window. Spam which used to charge calls advertises a product and provides a telephone number, so that if users call that number, they will receive a bill. Marketing research spam usually asks users to complete a questionnaire and send the information to a specified address. However, some advertising spam includes viruses that infect a user's system [10]. Usually, spam promotes a product or services and marks up the counter on the website [11].

**Nigerian letter**: Sometimes spammers use spam to lure money from the message recipient. The most common method is called the "Nigerian letter" (a huge amount of these messages come from Nigeria). Nigerian letters contain a text which talks about money that the recipient can get from sender under certain circumstances: for example, for helping the sender to send some money to organise documents. Luring money from victims is the purpose of this type of fraud [12].
**Phishing:** It is a scam using spam messages. Phishing is an attempt to defraud the recipient by taking a decisive information details such as credit card numbers, passwords or accessing email accounts. This kind of message usually disguised as an official letter from administration and asks the recipient to provide information by following the link provided by the spammer [13]. Often phishers (people who send phish emails) use emotive writing form or pretend to be licensed companies to gain the trust of their reality to receiver. Sometimes phishing spam includes get-rich schemes. It is a spam that proposes to spend a dollar to make millions often offers to participate in a pyramid scheme. This is not even mathematically possible. Often, only the organisers of the pyramid are getting money, and the rest of the members are just spending it [14].

**Sex-related spam:** It is one of the most popular and annoying. The porn industry brings enormous profits, so the moguls do not stint on advertising dirty and porn sites, and all kinds of indecent things like videos and adult products. These messages often contain explicit or hidden advertising of a certain porn site. Sex-related spam can cause severe damage to the psyche of children and adolescents, who have their own email addresses [15].

**Chain letters:** It is another spam that encourages reproducing letters and sending them to as many recipients as the user can find. It usually scares receiver with consequences if the conditions in the message are not met, or inspires by benefits that the user will get if the conditions are met. Usually letter starts with the well-defined phrase as “It is important! Please, do not remove, read it”. Chain letters do not cause a financial loss like a fraud pyramid schemes, and do not bring an emotional and psychological trauma like sex-related spam. However, they do fill the mailbox with useless messages and make the user pay for the traffic generated by spam [16].

### 2.2 The impact of spam

Mass mailing spam has a low cost. However, a significant amount of useless messages cause visible harm to recipients. First of all, it is all about time wasting on unnecessary screening and sorting important messages. Also, the Internet traffic is expensive, and the user has to pay for unwanted messages. It seems that spam can be beneficial for providers, as it increases the amount of traffic. But, in fact, providers also incur an additional cost due to the increased load of useless channels and tools. For this reason, providers have to spend resources on redundant equipment and systems protection from spam [17]. The aim of spam is to promote the message to the maximum number of recipients at a minimal cost. Moreover, the senders do not care about individual group of people. They only care about the amount of recipients.
Results from testing and evaluation of sending spam shows that it has a low impact. For example, during the advertising for the California Company, about 350 million notifications had been sent about a new pill based on natural products from 75,869 computers in 26 days. As a result, the company received only 28 orders in full [18].

2.3 Spam in numbers

The Royal Pingdom [19] produces articles such as “Internet in numbers” for each year. These articles present information about what happened with the Internet over that period. For example, in 2010 according to statistics from the Royal Pingdom, 262 billion spam messages were spent per day which showed that 89.1% of emails were spam.

According to the Symantec Intelligent Report produced by Symantec Corporation on November 2011, the global ratio of spam in email traffic fell by 3.7%, equating to one spam per 1.41 emails. Russia became the most spammed country, with a spam rate of 76.7%, and Saudi Arabia was the second most spammed, with 76.6% of email traffic blocked as spam. The spam level in UK was 69.5%. The most popular spam was pharmaceutical related, and the second most common related to watches or jewellery. The world phishing rate increased by 0.04%, taking the average to one in 302.0 emails (0.33%) that comprised some form of phishing attack. South Africa was most targeted for phishing attacks, with one in 96.2 emails identified as phishing. The UK was the second most targeted country, with one in 167.0 emails identified as phishing attacks. The global ratio of email-borne viruses in email traffic decreased to 0.03%, one in 255.8 emails (0.39%). The UK had the highest ratio of malicious emails so that one email out of 149.4 emails identified as malicious. Switzerland came second, and one in 185.6 emails identified as malicious [20].

Most of the Russian spam is about online advertising, product promotion and training workshops. Usually, these spam emails are received from free or hijacked personal unsolicited email accounts, without opt-out, and have randomized subjects to avoid being caught by spam filters. Regardless of the randomness, typical trait of spammers has been observed such that listing phone numbers in the email content as the only contact information instead of URL links.

Journalist Wikus Engelbrecht wrote an article, “Where spam is going in 2012 and what to do about it” [21], where he gave some statistics for 2012. According to his analysis, the most spammed countries are the United States and China. Malcolm James wrote a report on July 4, 2012 about spam [22]. He noted that 68 out of 100 emails were spam, and one in every 298 emails was a phishing email. The least spammed country was Japan with 65.1%. The most common spam messages are sex-related and pharmaceutical with 43% and 30.5% respectively.
In February 2013, Symantec Corporation [23] produced a report, where it showed that the phishing rate increased by 0.018%. In other words, one in 466.3 emails was identified as a phishing attack. The United States became the country where phishing is the most popular, compared to other countries. The percentage of phishing counted as 51.8%. The global ratio of email-borne viruses decreased to 0.11%. The global ratio of spam in email traffic went down to 65.9%, the sex-related and dating type of spam was the highest with 78.13%. The three email services, Yahoo, Microsoft and Google, are that used to send emails at a rate of 22%, 11% and 6%, respectively.

2.4 Spam filtering techniques

The following classes that measure protection was distinguished in relation to the concept of the Internet security problem:

1. Normative and legal. It is one of the best methods according to its power to protect against spam. Spammers send spam illegally, and it is subject to prosecution under the law. Without some aspects, such as working out the databases of the terminology, responsibilities and executive mechanisms, and coordination of actions with external control services, this law remains ink on paper.

2. Administrative and organisational. It is the simplest method: the user just should follow the rules to avoid being added to the spammers’ mailing list. The main rule are: “do not write primary email addresses everywhere, only give it to trusted friends and partners; create a secondary email address on free mail servers to use for additional work; create a more complicated password by mixing letters with numbers and signs; try to use a secondary email address for any forums or conferences by modifying it, so that people can always understand and write in a correct way; if you are an owner of the Internet source, do not provide an email address on the homepage, you should have a separate page for dealing with visitors, or use images to illustrate your email address” [24].

3. The software and hardware. This method provides software programs with different kinds of filters. Filters are used to classify the messages by type given in the settings of the email systems or mailboxes, for example by ignoring all the emails from trusted people and organisations, and filtering all the emails from unknown addresses to a separate folder.

Spam messages are usually markedly different from the usual correspondence. The common method of dealing with them is to examine the incoming mail flow to sort them out. There are software programs to automatically spot spam. It might be intended for the end users and also for servers. The main problem of automatic filtering is that it can mistakenly distinguish a legitimate message as spam. However, many programs and email services on the Internet offer the opportunity to save messages into a separate folder. So that people can still look for a legitimate message in the trash folder, but the lifetime of
messages in that folder limited. To prevent spam messages, both users and administrators use different powerful anti-spam technologies. The best-known and most popular technologies to avoid spam messages are blacklisting (DNSBL), mass control (DCC, Razor, Pyzor), scanning a message’s header, content filtering (Bayes) and greylisting.

**DNSBL:** Domain Name System Block List is one of the oldest anti-spam technologies. It blocks incoming messages sent from blacklisted IP-addresses. The benefit of this technology is that it filters emails from a suspicious source. The disadvantage is that it provides high rates of false positives, so it has to be applied with caution [25].

**Mass control:** Mass control (DCC, Razor, Pyzor) is a technology which identifies the flow of email messages, which are identical or slightly differ. This technology requires a tremendous amount of email messages to create the “mass” analyser. This technology is offered only by large companies to analyze significant amounts of email messages. The advantage of this technology is that, when it has been deployed, it is guaranteed to identify the mass newsletter. The disadvantages are that the “mass” distribution of messages may not be spam, and spammers can “breakthrough” protection with smart techniques. They use software to create a different text and, graphics for the message content. So “mass” control will not work in this case.

**Subject scanning:** Spammers build programs that generate spam and spread it instantly. However, they may incorrectly complete subject area, so that it no longer satisfies the conditions of email standards Request for Comments (RFC). RFC describes the structure of the header. According to the mistake on subject of the message provided method can detect spam messages. The advantage of this technology is that the process of identifying and filtering spam is transparent, standardized and fairly reliable. The disadvantage is that spammers can learn quickly, and error in the headers of spam is becoming less. The use of this technology will only hold no more than one-third of all spam.

**Content-based spam filtering:** Content filtering technology is one of the old technologies. Spam messages are checked for the presence of spam words, fragments of text, images and other typical spam characteristics. As a result of analysis, a text signature can be constructed or the “spamming weight” can be produced [26]. As content filtering based on content categorization, two methods of classification can be identified. First one has done using manually generating the rules. This method can be used when characteristics of the message are consistent, and features (words in the text) are well defined. The second method uses machine learning techniques. It requires the samples to learn. So that, it can adapt to any user and has more accuracy compare to static rule based method.
A Client-based Anti-spam Filter

**Greylisting:** It is the technology which temporarily rejects email messages. It is a good technology to solve the problem of spam. However, the disadvantage of this technology is that the recipient will receive the message with some delay. For many users, it is not a great solution [27].

### 2.5 Existing anti-spam programs

The modern IT industry offers a variety of solutions (anti-spam programs) to solve such a problem. The criterion of effectiveness of the anti-spam programs determine by the rate of hit-and-miss spam messages. In order to provide more details about how anti-spam programs work, their techniques and effectiveness, next step is to explore a well-known software program. Almost all the entire program works on the same principle.

**SpamAssassin:** It is a computer program that is used for email spam filtering based on content-matching rules [28]. It is a server-based program that combines different techniques, such as DNS-based and checksum based detection and Bayesian filtering. SpamAssassin is built using powerful and flexible language Perl, which allows using a combined score from several types of checks to determine whether a given message spam or not. The program is highly configurable and was incorporated with the mail server so that it automatically checks all mails. It includes a significant number of rules that needed to determine if the email is spam or not. The rules are based on regular expressions that are matched against the body or header fields of the message. However, this software is based on pattern recognition and it has side effects so it is possible to lose legitimate messages. SpamAssassin is a widely used email management program that is still improving its filtering techniques by using volunteers’ help [28].

**qSheff:** It is an improved content filter for the qmail queue e-server. The software program scans emails for spam, so that unsolicited spam messages are rejected before they go to the entire queue. So, after scanning the message the qmail queue is become enable. The process of catching emails before they join the queue decreases the email server load. qSheff has features responsible for a direct connection to ClamAv Antivirus daemon over socket, filtering by header or the body of the message. [29].

**DSPAM:** It is a content-based spam filter designed for multi-user enterprise systems. As it allows understanding the manners of the individual users, it can be called an adaptive filter. It checks the message content and learns what content the user will recognize as spam, without applying any list of rules. This approach provides high accuracy of catching spam messages. The main features of program are following: easy to use learning process, supporting a variety of storage implementation and being administratively free filtering
that written in C for speed, performance and scalability. DSPAM – supports many different message transfer agents (MTA), and can also be used as a stand-alone simple mail transfer protocol (SMTP) tool [30].

**VMware Zimbra:** Zimbra is a groupware email server based on organising emails [31]. It comes with built-in anti-spam filtering on the server, using open source tools, such as SpamAssassin and DSPAM. It helps on-going training allowing which allows organizations to optimize performance in their own environment. Therefore, the effectiveness of the software program is quite the same.

However, none of the existing technologies is a magic “silver bullet” against spam; a comprehensive solution does not exist. The most recent product uses different technologies in parallel. The efficiency of the product will not be strong enough if only one technology uses. The next chapter is going to introduce a new client-based anti-spam filter. The detailed knowledge about the work is also provided in the following chapters.
Chapter 3: Design

The first step before constructing any good software program, developers should write the software design of the program. Determining the design of the project before generating an actual code is vital, as it helps identify the requirements and, main functions of the constructed program. This chapter provides readers with information about the general structure of the developed program and the implementation tools that have been used to build it.

3.1 Software design

**Functional and non-functional requirements:** Designation of requirements is a significant step before building any software project. While reading all the resources that explain details about spam, its types and the tools to avoid it, the following functional and non-functional requirements were determined. First of all, the client-based anti-spam filter has to be built as an application so that any user will run it without difficulties. The next non-functional requirement is its configurability, so that users will monitor what is going on. Focusing on incoming messages and applying different anti-spam techniques concurrently are the functional requirements of the application. However, the important requirement of the program is to decrease the false positive rate, which is proportionally dependent on the number of lost legitimate messages.

**Activity diagram:** Before generating any codes and deciding the structure of the project, unified modelling language (UML) was used to create activity, domain and system class diagrams. It helped create a conceptual model of the anti-spam system. So, it was easy to understand, visualize and plan the project itself. The activity diagram (Figure 1) represents the expansion of the components of the system that has been built and shows its action in the use of coordinated sequential and parallel execution of processes. Moreover, it describes the general workflow of the software program that was going to be built [32].
Figure 1: The UML activity diagram of the system
Use case diagram: It was used to represent the relationship between actors and use cases so that it describes the software at a conceptual level as shown in Figure 2. The actors help to visualize the interaction with the system. Use cases were used to identify key actions of the anti-spam program [33].

Class Diagram: UML class diagram was used to describe the structure of the anti-spam filter by identifying appropriate classes, attributes, methods and relationship between those classes. As the field of a client-based anti-spam project is too vague, it was challenged to choose the exact area to implement. Therefore, starting from building a diagram before generating the code itself helped to establish a fundamental feature that has to be done during the project time. The main classes with attributes and methods of the project and their overall relationship are shown in Figure 3.
3.2 System architecture

After identifying all the behaviours of the application, a structure of the program has been constructed, where the key parts of the software program have been initialized. The structure of the system of a client-based anti-spam filter includes three main parts (Figure 4). The idea of the first part is to allow interaction between the user and system. It allows uploading a message or set of messages to check against spam. The second part is invisible to the user, called black-box, which includes all filtering techniques. So, the uploaded message goes through all the techniques that have been chosen by the user. The last part is responsible for maintaining the decision to the user and letting choosing the next step to make. The first and last parts of the project have been built by using a graphical user interface (GUI). The main part, where all the actions occur, is divided into three components. The first element is a front-end module, where electronic messages are parsed and tokenized. The second is a classifier module, where the email passes through several filtering methods and a probability of being spam is computed. There are two filters: one based on a Naive Bayesian classifier and another based on listing methodologies. Depending on the user’s decision the electronic message can be checked through either one or both of the provided filters concurrently. According to the information computed in the second component, the last element finally makes a decision and sends the result.

![Diagram of system structure](https://via.placeholder.com/150)

*Figure 4: The structure of the system*
3.3 Implementation technologies and APIs

While implementing the project, two environments were tried such as Eclipse and NetBeans. They have common functions, which allow programmers to build java application in a way they want. The given project was built by using NetBeans IDE (Integrated Development Environment). The intention for using such environment was made according to its advantages while coding in Java and building GUI to illustrate the application. NetBeans is more functional and has Maven support, which allows guiding a project by gathering principles like specification, implementation and unit testing. It stores the unit test code in a separate subdirectory. Configuration of Netbeans is intuitive and easy to follow, compared to Eclipse’s. Also, NetBeans provides a workflow, which gives an opportunity to provide guidance and issue tracking, and suggests a good layout of directories to navigate through them. The environment has better language support, which allows building an application using different programming languages like Java, C/C#, Perl and other.

The conceptual idea of the project is to build software that identifies spam messages using different filters. The application is helpful when the power of the result is high, and the time complexity to execute the code is short. To meet such criteria, Java is the best suited programming language. It allows writing a complicated code in a small number of lines, compared to C language. The benefit of using java is that it is supported on most platforms such as Linux, Microsoft. The compiler for the language is built better than for C language as it displays the error with an appropriate message, as long as the C compiler displays only the message like “Segmentation fault”. A debugging the code provides with the necessary information. Also, garbage collection is already built into the java language. As the project primarily based on storing the data, Hash map was used to reduce the use of memory space. The availability of functions for hash map was one of the reasons of using Java language.

After determining the performance of the program and choosing implementation tools, the next step was to implement into the real world. The next chapter is going to provide detailed knowledge about how the project was built by providing the detailed description of each algorithm that has been implemented. The process of generating a user interface is also demonstrated in chapter four.
Chapter 4: Implementation

Researchers have proposed many spam resistance approaches, including white and black lists, statistical filtering, sender authentication and network analysis. A single commercial product often employs many of these approaches in parallel. The project has been made according to this statement. In order to fill identified requirements that were mentioned in chapter one, software program was implemented by mixing different methodologies such as context analysis, Bayesian classifier, greylisting and blacklisting. As the effect, it provides users with two different filters to defeat spam. The general algorithm for a client-based anti-spam filter is shown in Figure 5. This chapter provides detailed information of implementation techniques.

Figure 5: The workflow of the system
4.1 A filter based on Naive Bayesian approach

While reading all the research papers about existing anti-spam techniques, and their advantages and disadvantages the decision was made to use a statistical text classifier based on Bayes’ theorem. The Naive Bayesian classifier is a learning-based statistical filter [34], which classifies messages into one of two classes, spam and non-spam messages, depending on its values of words. The value of word is calculated using a training dataset provided by the user. After that, for each incoming message, the given technique calculates a score based on spam and non-spam values, and according to the score, a decision is made about into which class (spam or non-spam) the given message has to be classified.

First of all, to create a classifier with similar behaviour, a class was constructed which is responsible for an individual word in the message. The class Word.java contains a method that calculates the probability of each word. It depends on the total number of spam and non-spam messages and the total number of appearances of each it in spam messages. Figure 6 represents the pseudo-code of described function. For non-spam messages, the same approach has been applied, but instead of searching in spam, it was changed as non-spam messages.

```
If word is in spam
    increase the countSpam by 1;
If word is in non-spam
    increase the countNonSpam by 1;
```

*Figure 6: The pseudo-code of counting appearance of word*

The main idea of Bayesian filtering says that the chance of an event occurring in the future may infer from the previous occurrences of that event [34]. To find out whether a word appeared in either a spam message or non-spam message the filter has to be trained with a dataset for spam and non-spam messages, respectively. The dataset for each message type was collected from personal and friends’ email accounts. Also, knowledge about different types of spam messages that mentioned in chapter two was used to create a dataset. According to dataset many spams can be identified. The pseudo code of the method to train the filter is shown in Figure 7. The same approach has been used to train non-spam messages, but instead of counting the total number of spam messages, the total number of non-spam messages was counted. Then, in order to classify the message into one of the classes, filter should check for the presence of words from text in any of two classes. Hash map was used to determine whether a given word is in a spam message or not as it has a constant time O(1) for searching using a word as a key.
The same steps repeated to train another dataset for non-spam messages. However, now the number of appearances of each word in the non-spam messages has been increased. By using the following formulas (1 and 2), the total probability of being spam $p(s)$ and the total probability of being non-spam $p(ns)$ have been found.

$$p(s) = \frac{\text{countSpam}}{\text{totalNumberOfSpam}}$$  \hspace{1cm} \text{(formula 1)}

$$p(ns) = \frac{2 \times \text{countNonSpam}}{\text{totalNumberOfNonSpam}}$$  \hspace{1cm} \text{(formula 2)}

The count number for non-spam messages was multiplied by 2. This was done to equalize the power of words in non-spam messages. Assumption was made according to information gained from research papers that, almost all the words in legitimate messages are also presented in spam messages. There are only a small number of words which are included only to legitimate messages.

After training the Bayesian technology, the filter was assessed. Tested email is loaded and tokenized and the 15 most interesting words are chosen and stored into a new array. Choosing the most interesting 15 words was done by taking the difference between 0.5 and the probability of the word in the table. The purpose for taking exactly this number was made by assuming that the possibility of being given word in a spam message is the same as in non-spam messages, which equals to 50%. Formula 3 has been used to calculate interesting rate:
A Client-based Anti-spam Filter

\[ interesting = |0.5 - p(s|w)| \]  \hspace{1cm} (formula 3)

where, \( p(s|w) \) is the probability of message being spam, given that the word is in it.

According to the 15 words chosen from the actual message, the final probability of the message being spam, given that the word is in it, is calculated. The idea of Bayes’ theorem helps to do so as shown in formula 4:

\[
p(s|w) = \frac{p(w|s) * p(s)}{p(w|s) * p(s) + p(w|ns) * p(ns)} \]  \hspace{1cm} (formula 4)

where, \( p(s) \) is the overall probability that message is spam (see formula 1); \( p(w|s) \) is the probability that the word \( w \) shown in spam messages; \( p(ns) \) is the probability that the message is non-spam (see formula 2); \( p(w|ns) \) is the probability that the word \( w \) appears in non-spam messages. The written code is provided in Figure 8.

```java
private static float calculate(ArrayList interesting) {
    float pposproduct = 1.0f;
    float pnegproduct = 1.0f;
    for (int i = 0; i < interesting.size(); i++) {
        Word w = (Word) interesting.get(i);
        pposproduct *= w.getPSpam();
        pnegproduct *= (1.0f - w.getPSpam());
    } //for

    float pspam = pposproduct / (pposproduct + pnegproduct);
    return pspam;
} //calculate
```

*Figure 8: Implementing Bayes’ theorem to calculate final probability*

The next thing to classify a message into one of two classes (spam and non-spam), was assigning the value for the threshold:

\[
\frac{p(ns|w)}{p(s|w)} > t \]  \hspace{1cm} (formula 5)

where, \( t \) is a threshold. The value of threshold has been chosen after certain experiments to be 0.49. This number helps to reduce the rate of false positive (rejecting good messages) and false negative (missing spam messages). For example, there are words which appear in
spam messages almost the same times as in non-spam messages, like “the”, “money”, “home”, “solution”, etc.

The Bayesian calculation was used for the single keyword sets. However, to improve the accuracy of filter a keyword context set has been added. In another word, the first Naive Bayesian filter was implemented for a single keyword set and then the keyword is mapped to a keyword context. The context of keyword is a set that contains all the keywords except itself. For example, a keyword “win” can have a context as [vvin, w1n, vvii_n]. The identified keyword is used to assign a Bayesian probability related score. The keyword contexts are compared to the set of existing keywords, to get a context matching percentage. Next, to improve the accuracy, the keyword context score was added in order to improve the Bayesian score. Spam score calculated with respect to the matches spam keyword contexts that found in the existing database of keywords:

\[
S_{\text{total}} = \sum_{i,j=1}^{\ldots,n} S(\text{spamKW}_{ij} \cdot \text{weightKW}_{ij}) + CS(\text{spamKW}_{ij} \cdot \text{weightKW}_{ij})
\]

(Formula 6)

where, \(S_{\text{total}}\) is the total spam score; \(S(\text{spamKW}_{ij} \cdot \text{weightKW}_{ij})\) is the weighted Bayesian score for keyword; \(CS(\text{spamKW}_{ij} \cdot \text{weightKW}_{ij})\) is the weighted Bayesian score for matching keyword context.

**Challenges:** The challenges form during constructing the filter was dealing with memory and identifying keyword contexts. Training the filter with a huge number of messages requires significant free spaces in the memory to store all the possible words. Decision to overcome this issue was using a database, but it took a time to access to the database. So that, at the end of experiments, a decision was changed to store all the words in the hash table, which allows saving them in the main memory. Accessing information from the main memory does not take too much time, compared to querying information from the database.
4.2 A filter based on listing methodologies

The second filter was designed by mixing particular listing techniques like greylisting, blacklisting and whitelisting. The main objective of greylisting is to protect against spam by temporarily rejecting the incoming messages from unknown sources. Consequently, the idea of blacklisting is to reject all the messages from the sender that is already in the black list of the recipient. However, the whitelisting has an idea which is opposed to blacklisting, so that, the recipient will receive all the incoming messages from the sender already in the white list, and all other messages will go to the trash folder. These techniques are quite similar, so implementing them together as one listing methodologies increases the efficiency of the filter. The pseudo-code is shown in Figure 9 was used to create a combined algorithm.

```
receive email; 
get IP-address of the sender; 
if it is in white list 
    accept email; 
else if it is in black list 
    reject email; 
else wait for 5 minutes 
    if request received again 
        accept email; 
        add IP-address to the white list; 
    else reject email 
        add IP-address to the black list; 
```

*Figure 9: A pseudo-code of combined listing algorithm*

However, as already mentioned, the greylisting technology works on a message transfer agent (MTA) level, which requires a service connection. The concept of temporary rejecting messages and storing them into a separate folder was applied to the second filter. Also, an essential feature of black and white listing technologies was taken. So that, IP-addresses stores into a list, not just for some certain time, but forever unless the user will delete or replace them.

There are some reasons to have both black and white lists. The first argument is to minimize an execution time of process. The second reason is to increase the efficiency of spam detecting. The third reason is to reduce the amount of inadvertently blocking legitimate messages. An assumption was made that most people will receive legitimate messages only from already known contacts, rather than a new one. Moreover, almost all
people receive the advertising or other messages from the same IP-address, even if spammers try to change their existing email address. So that, the application will spend less time to check for spam messages, by skipping all unnecessary steps.

**Challenges:** The following difficulties have been met while building a filter. First of all, collecting all the relevant data took a huge amount of time. The next problem was storing the accurate details for each message. The greylisting methods involve working with triplets, where triplet is a collection of information like IP-address, sender’s email address and recipient’s email address. Moreover, finding a place to save a substantial number of dataset was not easy. In beginning of implementation, all the information stored in the database system by creating appropriate tables. However, the same problem as for the first filter was appeared while building the second filter. Accessing the database and querying any information would take more time, than working with a file from computer memory. The decision was made to produce triplet figures in a table and store them in a file. A big number of senders increased the overall length of the file, which took a significant free space of the main memory. No user would allocate memory space to store perhaps useless information. So, the third problem came out, which was to keep only essential information to save as much memory spaces as possible. To do so, only information that provides with the IP-address of the sender has been taken as this information is more beneficial for spam catching. Finally, an array was created where the IP-address has been stored, and array was written into a single text file. This answer came up in order to improve the time complexity of the filter.

First of all, the knowledge about the structure of the message was used to get the IP-addresses of the messages and keep them in a separate file. Java provides a powerful feature which is a structural pattern matching feature which is available through individual classes like Pattern and Matcher. Pattern matching technique is used to control where the provided token is present in some pattern. Pattern recompiles the regular expression to perform efficiently. Also, it makes sure that the expression can be reused by Matcher. All the objects of Matcher are created by the Pattern. No time will spend to compile regular expression. Pattern.compile() method will make the job for us. Also, Matcher provides methods such as find(), matches(), etc, that allow scanning in a short time [35]. So, the described method was used to get the IP-address. Figure 10 represents a part of the actual code where described written getIP() method.
Figure 10: Part of the actual code with Pattern and Matcher

The quality of the second developed filter is lower, compared to the first one. It was defined from the false positive rate (losing legitimate messages). However, when the generated software program (application) runs two filters at a time, it increases the efficiency of catching spam messages without killing legitimate ones. Detailed information about how the two filters work alone and concurrently under different circumstances is introduced in the testing and evaluation chapter.
4.3 Improvements

At the beginning of implementation, the program was planned to examine only one text file at a time. However, users would be tired to do the same step for each message that has to be checked. To meet user's expectation, the new program works not only for one exact message, but also for collection of messages stored in a folder. The method shown in Figure 11 has been added to complete scanning technology for each message in the provided folder.

```java
this.words = new HashMap();
countGood = 0;
countBad = 0;

File goodFileDirectory = new File("D:\spamfiltering\AntiSpamFiltering\good\"); 
File[] goodFilesInDirectory = goodFileDirectory.listFiles();

File badFileDirectory = new File("D:\spamfiltering\AntiSpamFiltering\spam\"); 
File[] badFilesInDirectory = badFileDirectory.listFiles();

File testFile = new File(test);
File[] testFilesInDirectory = testFile.listFiles();
```

*Figure 11: The method to list all the files from a folder*

A user interface has been changed according to the new implementation. Figure 12 illustrates the previous version of the application, and Figure 13 shows the current interface of the application.
Figure 12: Old interface

Figure 13: Current interface
4.4 UI Design

As the project is called “A client-based anti-spam filter”, the idea was to achieve a user-friendly client-based application, where the user is essential. To do so, the program was constructed by using a graphical user interface (GUI). Netbeans IDE provides an opportunity to make GUI and join the classes to it using Java code. Also, it allows choosing different colours so that the software attracts the users’ attention.

Arranging the position of buttons, text fields and labels were made after effective feedback that was given by testers. According to the research papers and testers’ opinion the final version of user interface design of the program has been constructed as shown in Figure 14. The purple coloration is chosen as the main buttons’ tint so that it gives an indication that this program was made by a student of the University of Manchester.

When the program runs, it opens the window, where user uploads a message or set of messages to check. The given message is displayed on the text field next to the upload button. The second step is to provide the user to choose either one or two filters that have been provided bellow. User presses “check button”, which passes the given message/messages through filter/filters after choosing the filters. When the checking process is finished, pops up a new window with the result message and all statistics is provided on the main window of application in the table form. A further action can be varied as it depends on the user. The user can either choose to quite the program or to clear the text fields and upload a new message or set of messages.
Chapter 5: Result

A client-based anti-spam filter was aimed to achieve the same or even better efficiency, compared to other existing anti-spam tools in a real world. This chapter is going to start performance of the final product that was implemented during the project time. The efficiency of the program was found under certain conditions and judged by the rate of false positive values (the detailed information provided in the next chapter). How the spam works in general, is shown in Table 1. The key steps were underlined with explanatory materials.

Table 1: Workflow of the application

<table>
<thead>
<tr>
<th>Step</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Run the application</td>
<td><img src="image" alt="A client-based anti-spam filter" /></td>
</tr>
<tr>
<td>2. Upload file/files to work with</td>
<td>![Envelope]</td>
</tr>
<tr>
<td>3. Choose filters to check through</td>
<td><img src="image" alt="Naive Bayesian algorithm" /> <img src="image" alt="Listing algorithm" /></td>
</tr>
<tr>
<td>4. Press check button</td>
<td>![Next button]</td>
</tr>
<tr>
<td>5. Get the answer</td>
<td><img src="image" alt="Files have been checked for spam" /> <img src="image" alt="Message is NOT spam" /> <img src="image" alt="Message is spam" /></td>
</tr>
<tr>
<td>6. Quite the application</td>
<td>![Power button]</td>
</tr>
</tbody>
</table>
The anti-spam filter examines the content of the provided message, and according to the implementation of each algorithm, it labels as spam or non-spam message. Usually existing anti-spam programs activate without any help from the client, and such filters are called as server-based anti-spam filters. However, this program activates when the user will press the running button. It is useful because some messages in the user's mailbox can be suspicious. Availability of double checking the received electronic messages are provided by software developed during project time.

Several ways to improve configurability of the program were made. The first thing is to allow the user to work with different data types. As mentioned in implementation chapter the user can check one or more messages. It happens by allowing uploading either a file or a folder. Figure 15 illustrates the dialog window where the user can choose different data to examine.

![Open dialog window](image)

*Figure 15: “Open” dialog window*

The text field was provided to display the message context (Figure 16). However, it only works when the user checks one message at a time. Figure 17 illustrates situation when a directory has been chosen to check through the filter.
The next way to make the application customizable is to provide repeating the process of checking as many times as the user wants to do it. The button illustrated bellow “upload button” is called “clear”. It is responsible for clearing all the written information from text fields, option choices and result table. It was used to provide strong feelings to the user that the program running from the beginning and the old data will not affect the result of the new execution. Finally, choosing an option for the filters to check through is also made in order to improve the configurability of the application. All the screen shots of the running process under in different situations are provided in the appendix provided by this report.

The result of classifying the checking message was done by using the logical operator AND. The operation of AND is shown in Table 2, and the actual result from the program is shown in Figure 18.
### Table 2: Behaviour of logical operator AND

<table>
<thead>
<tr>
<th>Filter1</th>
<th>Filter2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

![Result part of the application](image)

*Figure 18: Result part of the application*
Chapter 6: Testing and evaluation

6.1 Testing

The testing process is an essential part while building a new software program [36]. Testing is used to determine the behaviour of the program and examine how certain parts respond to the user's query. Since testing is an irreplaceable component, it is used to examine a full program for a bug and determine whether the program has met the requirements or not. Also, testing is required to confirm the end-user the quality of the product, and that the final version of the product is doing the right thing [37]. In this chapter tables and diagrams were used in order to provide the capability of the system under different test cases.

To test the project 500 spam and 500 non-spam electronic messages have been used. Testing was done by using three different sets of messages:

- First set, where all the messages are spam;
- Second set, where all the messages are non-spam;
- Third set, where messages are mixed (50% of spam and 50% of non-spam).

Accuracy of the testing was calculated using a confusion matrix, which allows a visualization of the performance of an algorithm [38]. The following Table 3 represents the relationship between actual and predictive values that describe confusion matrix:

Table 3: Confusion matrix

<table>
<thead>
<tr>
<th>Actual value</th>
<th>Predicted value</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spam</td>
<td>Non-Spam</td>
<td></td>
</tr>
<tr>
<td>Spam</td>
<td>True Positive</td>
<td>False Negative</td>
<td></td>
</tr>
<tr>
<td>Non-Spam</td>
<td>False Positive</td>
<td>True Negative</td>
<td></td>
</tr>
</tbody>
</table>

In the table, actual value represents the number of spam and non-spam messages after the experiment, and predicted value represents the amount of spam and non-spam messages that should be checked. So, the first row is the positive prediction value and the second row
are the negative prediction value. To calculate accuracy of the result following formula has
been used:

\[
acc = \frac{\sum TP + TN}{\sum TP + FP + FN + TN}
\]

where, \( acc \) is the accuracy, \( TP \) is the total value of true positive, \( TN \) is the total number of
true negative, \( FP \) is the total number of false positive and \( FN \) is the total number of false
negative.

6.1.1 Test case 1

Testing was started by taking a small amount of messages as 1 and increased till 500
electronic messages. The first test case was to verify the accuracy of filter while all the
testing messages are spam. As only spam messages were tested for correct and incorrect
classification messages, the general formula provided to find accuracy can be reorganised
as follows:

\[
acc = \frac{\sum TP}{\sum TP + FP}
\]

where, \( TP \) is the value of correctly classified spam messages and \( FP \) is the value of spam
messages that have been classified as non-spam. The following three tables show the effect
of the experiment with different filters.

Table 4: Result of Naive Bayesian classifier

<table>
<thead>
<tr>
<th>Messages</th>
<th>TP</th>
<th>FP</th>
<th>Accuracy, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>49</td>
<td>1</td>
<td>98</td>
</tr>
<tr>
<td>100</td>
<td>97</td>
<td>3</td>
<td>97</td>
</tr>
<tr>
<td>200</td>
<td>194</td>
<td>6</td>
<td>97</td>
</tr>
<tr>
<td>400</td>
<td>387</td>
<td>13</td>
<td>96.75</td>
</tr>
<tr>
<td>500</td>
<td>480</td>
<td>20</td>
<td>96</td>
</tr>
</tbody>
</table>
The average accuracy of Bayesian filter is 98.2%. However, the average accuracy provided by listing methodologies is 95.1%, which is lower than statistical filtering. The average accuracy of the third table is 98.5%. The experiment result shows that to get higher accuracy of catching spam messages without losing legitimate messages two filters should be used concurrently.
6.1.2 Test case 2

The result of the following testing is illustrated by using diagrams shown in Figure 19, and almost the same behaviour of the filters was observed during this test case.

![Image](image.png)

*Figure 19: Result part of the application*

6.1.3 Test case 3

In order to verify the accuracy of the third experiment, the values of true positive, false positive and false negative were considered. Also, the general formula of recognition accuracy was used, which introduced at the beginning of this chapter. In this chapter, the accuracy of each filter has not displayed as it gave almost the same result. However, to determine the level of performance of the final project result of testing by using combined filters is illustrated in Table 7. The training data was chosen by taking half of the total number of messages to be spam, and another half to be non-spam.

<table>
<thead>
<tr>
<th>Messages</th>
<th>TP</th>
<th>FN</th>
<th>FP</th>
<th>Accuracy, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>100,00</td>
</tr>
<tr>
<td>100</td>
<td>94</td>
<td>3</td>
<td>3</td>
<td>94,00</td>
</tr>
<tr>
<td>300</td>
<td>293</td>
<td>5</td>
<td>2</td>
<td>97,67</td>
</tr>
<tr>
<td>500</td>
<td>488</td>
<td>7</td>
<td>5</td>
<td>97,60</td>
</tr>
</tbody>
</table>

*Table 7: Testing spam and non-spam messages using combined filters*
6.2 Evaluation

Evaluation is another step that determines the deliverability of the project. It helps the software developer to determine the satisfaction of end-users with the provided software system. Because of the evaluation process the developer identifies other required developments. The main purpose of the evaluation is to check whether the checking project fulfils certain criteria and to evaluate the value of it [39].

At the beginning of the project, three main approaches such as systematic, issue-oriented and user-driven has been chosen to explain the program for consistency, expandability and configurability. All the evaluation process has been done using an external software program and real end-users. Consistency of the system was checked by running some test cases. Expandability of the system was tested by sticking to the concept of low coupling and high cohesion. So, if someone else wants to develop the project, an additional code will not affect the previous work. Several feedbacks were received from people in order to determine the user-interface for easy interaction. Also, the project was tested for time complexity so that the client will not wait for long until a huge number of messages will pass the filter and the objective results will appear on the monitor. As mentioned in chapter three, the colours that used to describe the message and represent action that have been chosen owing to the suggestion of testers. The overall form of the developed program can be seen in appendix.

According to the analysis and evaluation results, assumption was made that the final year project was completed successfully and met all the requirements that have been defined at the start of the development process in spite of difficulties during the building period.
Chapter 7: Conclusions

In the next few years when everything will be computerised, and the primary data source will be Internet, a problem with spam will become irresistible. In order to participate to solve the problem with spam now, so not too late, the project to create a new client-based anti-spam filter proposed. The aims and objectives of the project, which achieved throughout the course, defined at the very first stage of the process. To collect all the information, the research work involved a careful study on the different filtering algorithms and existing anti-spam tools. All the information related to spam and spam categories that were described in chapter two were collected during research time. These large scale research papers and existing software programs are one of the sources of inspiration behind this project work.

During the practical work, a user-friendly and configurable client-based anti-spam filter program was developed. The application was built using Object Oriented Programming Language Java. The complete operating instructions of the program were given in chapters four, five and six. The functionality of the software product was examined by performing some basic and advanced tests. The system passed all the test cases, and thus accomplished the final stage of the project.

The skills and the experience gained while developing an application from scratch were necessary as it was a first step to software program development. The amount of knowledge gained by performing substantial work was accomplished, and it cannot be achieved by just reading books and articles. The project helped to develop and improve the understanding of OOP language Java and Netbeans IDE.

The project was a remarkably complex task. Developing a software program is incredibly difficult. Also, a creating perfect customization possibility is the problem on its own. Countless programming skills are required to build a real masterpiece. However, working on such project provided with excellent experience. The project planning and project management skills helped to complete the entire project. Use of Agile approach helped to meet the deadlines. The whole project was divided into several iterations. Each iteration was completed by completing four phases: inception, where the idea of work was identified; elaboration, where architecture of the part of the system is designed; construction, where existing code is implemented; transition, where the developed part of the project is validated.

However, there are still some parts that can be improved: for example, adding additional filtering techniques or changing aspects of the existing ones. The changes such as incrementing or decrementing the number of interesting words of the message and reorganizing the formula for calculating interesting rate can be done later. The application
can be regenerated to build into the actual email account, so it provides online and offline filtering mood. In order to detect spam messages, that contains only images more anti-spam techniques can be added to the software program. So that high efficiency of catching spam can be guaranteed.

Overall, the project was completed successfully as the main objectives of developing a client-based anti-spam filter program were achieved. The program satisfies functional and non-functional requirements. The interface of the program kept as uncluttered as possible. To sum up, this report was written to describe the effort taken to develop and implement the client-based anti-spam filter software program.
References


A Client-based Anti-spam Filter


Figure 1: Detecting a spam message using filter 1.
A Client-based Anti-spam Filter

Figure 2: Detecting a spam message using filter 2.
Figure 3: Detecting a spam message using both 1 and 2 filters:
Figure 4: Filtering a set of messages to detect spam using filter 1.
Figure 5: Filtering a set of messages to detect spam using filter 2.
Figure 6: Filtering a set of messages to detect spam using both 1 and 2 filters.