Sudan University of Science & Technology
College of Graduate Studies

A Personalized Arabic Spam Detection Model

نموذج عربي مشخصن لاكتشاف رسائل البريد الالكترونية غير المرغوبة

A Thesis Submitted in Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science

By:
Asma Ibrahim Gamar Eldeen Mohammad

Supervised by:
Prof. Izzeldain Mohammed Osman

May 2014

ABSTRACT

In a free multicultural society a spam message is different from one user to another, i.e. certain content may be acceptable to one user but not be acceptable to another. So what is “unwanted” by one user may be liked by another user, what is classified as spam by one user at sometime may not be classified by the same user at other time. Therefore, there is a need to extend the standard spam filters to incorporate the different interests of the users and the changing interests of each user.

In this thesis an attempt is made to extend the spam detection to follow the liking of the user. This is termed personalized spam detection. Thus the main objective of this work is to design a user personalized algorithm to detect English spam and modify it according to the complexity of Arabic language to detect Arabic and mixed (Arabic and English) spam emails.

A dataset of Arabic emails which includes spam and non-spam is built. The data set is used to train Naïve Bayesian classifier to build Arabic spam detection model. Cross validation experiments are used to evaluate the model.

A personalized spam detection web based, Permail, is developed and used for comparison against the spam filtering capabilities of Microsoft Hotmail, Google Gmail, and Yahoo Mail and to determine the effectiveness of spam filtering for each provider. The criteria used in the comparison are the quantity and percentage of spam in the Inbox.

In this work three models are presented, the first one is an English spam detection model which uses a Naïve Bayesian algorithm where the model is trained using a large corpus of spam and non-spam messages and then tested using a standard dataset (From the Second Conference on Email and Anti-Spam CEAS 2005, Stanford University, Palo Alto, CA). The results are comparable to those obtained from other models. The model is then extended and modified to handle second model of Arabic and mixed (English and Arabic) data model. It is then tested against the Arabic corpus. A personalized web based spam detection system which was developed to provide a more personalized mail system to filter spam emails. Third model is personalized mail system (Permail). Which is classify spam message

based on the behavioral of each user and it can provide a more personalized mail system to filter spam emails.

The result of comparing performance of three classification techniques, Decision Tree J48, ZeroR, and Logistic Regression with the proposed Arabic spam detection shows the success criteria for text classification have significantly increased by using the proposed spam detection model.

The result of using the corpus of the body of the message is better than that of the subject. The result of comparing the web based spam detection system with three known mail systems showed that the proposed system is the best one.

المستخلص

في المجتمع المتعدد الثقافات إن رسالة البريد الالكتروني المزعج تختلف من مستخدم لآخر. فمن المحتوى ما يكون مقبولا لمستخدم لا يكون مقبولا لآخر. حتى ما هو "غير المرغوب فيه" من قبل مستخدم واحد قد يكون محبوبا من قبل مستخدم آخر، أي ما تصنف على أنها رسائل غير مرغوبة من قبل مستخدم في وقت ما قد لا يتم تصنيفها من قبل نفس المستخدم في وقت آخر. ولذلك فهناك حاجة إلى دمج المرشحات القياسية لتتماشى مع رغبات المستخدمين. في هذه الأطروحة تم إجراء محاولة لتطوير نموذج لكشف البريد الغير مرغوب لمتابعة الرغبات المتغيرة للمستخدم. ويطلق على ذلك كشف الرسائل المزعجة المشخصن

في هذه الأطروحة تم إجراء محاولة لتطوير نموذج لكشف البريد الالكتروني غير مرغوب لمتابعة الرغبات المتغيرة للمستخدم. ويطلق على ذلك كشف الرسائل غير المرغوبة المشخصن. الهدف الرئيسي من هذا العمل هو بناء نظام مشخصن للمستخدم لكشف رسائل البريد الالكتروني غير المرغوبة الإنجليزية وتعديله وفقا لمدى تعقيد اللغة العربية للكشف عن رسائل البريد الإلكتروني غير المرغوب العربية والمختلطة (العربية والإنجليزية).

تم تجميع رسائل بريد الكتروني عربية واستخدمت لاختبار نموذج الكشف عن البريد الالكتروني غبر المرغوب العربية. تم استخدام مجموعة الرسائل لتدريب مصنف (Naive Bayes) لبناء النموذج العربي للكشف عن البريد الالكتروني غير المرغوب. وتستخدم تجارب التحقق من صحة هذا النموذج.

تم تطوير واستخدام نظام الكشف المشخصن عن البريد المزعج على الوب ، Permail ، للمقارنة ضد قدرات مرشحات البريد المزعج من (Microsoft Hotmail, Google Gmail, Yahoo Mail)، لتحديد مدى فعالية ترشيح البريد المزعج لكل منتج . كانت المعايير المستخدمة في المقارنة كمية و نسبة البريد المزعج في قائمة البريد الوارد.

في هذا البحث يتم عرض ثلاثة نماذج ، الأول هو نموذج الكشف عن البريد المزعج للرسائل الانجليزية الذي يستخدم خوارزمية (Naïve Bayesian) حيث تم تدريب النموذج باستخدام مجموعة من رسائل البريد المزعج وغير المرغوبة ومن ثم اختبارها باستخدام مجموعة بيانات قياسية (من المؤتمر الثاني على البريد الالكتروني و مكافحة البريد المزعج CEAS عام 2005، جامعة ستانفورد ، بالو ألتو ، كاليفورنيا) . وكانت النتائج مماثلة لتلك التي تم الحصول عليها من نماذج أخرى . ثم جرى تطوير و تعديل نموذج للتعامل مع النموذج الثاني لرسائل الغة العربية و الرسائل المختلطة (العربية والإنجليزية). ثم تم اختباره ضد مجموعة من الرسائل العربية.تم تصميم النموذج الثالث وهو نظام الكشف المشخصن للبريد المزعج على الشبكة العالمية (Permail) والذي تم تطويره لتوفير نظام بريد أكثر تخصيصا لتصفية رسائل البريد الإلكتروني غير مرغوب فيها.

نتيجة لمقارنة أداء ثلاث تقنيات التصنيف، (Decision Tree J48, ZeroR, and Logistic Regression) مع نموذج الكشف عن البريد المزعج العربي المقترح اتضح ان معايير النجاح

لتصنيف النص قد زادت بشكل كبير عن طريق استخدام النموذج المقترح للكشف عن البريد المزعج. وقد كانت نتيجة استخدام مجموعة بيانات نص الرسالة أفضل من استخدام موضوع الرسالة.

اظهرت نتيجة استخدام مجموعة بيانات علي نص الرسالة هو افضل من موضوع الرسالة. وأظهرت نتيجة المقارنة بين نظام الكشف عن البريد المزعج على الشبكة العالمية مع ثلاثة من أنظمة البريد المعروف أن النظام المقترح كان الأفضل.

ACKNOWLEDGEMENTS

Thanks to Allah for giving me the power and help to accomplish this research. Without the grace of Allah, I was not able to accomplish this work.

I would like to express my thanks to my supervisor, Prof Izz Eden, for his advice, support, guidance, and assistance throughout the work of this research.

The most special thanks go to my husband, he gave me his unconditional support and love through all this long process, I also extend my thanks to all my family members for their motivation and support.

DEDICATION

To My Big Family and My Children Shreef & Asail

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SPAM	Unsolicited bulk email.
SMTP	Simple Mail Transfer Protocol.
POP	Post Office Protocol.
IMAP	Internet Message Access Protocol.
DEC	Digital Equipment Corporation.
WEKA	Waikato Environment for Knowledge Analysis.
UTF	Unicode Transformation Formats.
MatLab	Matrix Laboratory.
GUI	Graphical User Interface.
MTA	Mail Transfer Agent.
ISP	Internet Service Provider.
URL	Uniform Resource Locator.
NB	Naïve Bayesian.
SVM	Supported Vector Machines.
NNet	Neural Network.
kNN	k-Nearest-Neighbor.
SFFS	Sequential Forward Floating Selection.
MLE	Maximum Likelihood Estimate.
OCR	Optical Character Recognition.
NL	Natural Language.
NLP	Natural Language Process.
NLTK	Natural Language Toolkit.
IDLE	Interactive DeveLopment Environment.
WSIS	World Sumiton Information Society.
IM	Instant Messaging.
PCA	Principal Component Analysis.
ISRI	Information Science Research Institute.
IR	Information Retrieval.
DF	Document Frequency.
TF- IDF	Term Frequency - Inverse Document Frequency.
CSV	Comma Separated Value.
Spim	Spam delivered through IM.
SMS	Unsolicited text messages.
TREC	Text REtrieval Conference.
ARPANET	Advanced Research Projects Agency Network.
ODBC	Open Database Connectivity.
CLAWS	Constituent Likelihood Automatic Word-tagging System.
HPSG	Head-driven Phrase Structure Grammar.

MSA	Modern Standard Arabic.
ACA	Arabic Content Analysis.
AEC	Arabic Email Corpus.
ASP	Active Server Pages.
CA	Content Analysis.
FB	Flexible Bayesian.
ICA	International Corpus of Arabic.
SEO	Search Engine Optimizations.
SPF	Sender Policy Framework.
WCSIT	World of Computer Science and Information Technology.
ASDM	Arabic Spam Detection Model.
CAPTCHA	Completely Automated Public Turing-test to tell Computer and Humans Apart.
CSS	Cascade Style Sheet.
DKIM	DomainKeys Identified Mail.
SERP	Search Engine Results Page.
SUST	Sudan University of Science and Technology.
PASDP	A Personalized Arabic spam Detection Project.

CHAPTER 1: INTRODUCTION

CHAPTER ONE
INTRODUCTION

1.1 Motivation

To understand the problem of spam, we must first establish common definitions, explain why the problem exists, discuss limitations of current solutions, and make suggestions of work that can lead to solutions that minimize the effect of the spam problem.

Email is a method of sending and receiving messages over electronic communication systems such as the internet. The modern day protocol for sending email is the Simple Mail Transfer Protocol (SMTP), proposed in 1982[1]. The most commonly used protocols for email retrieval by client programs are the Post Office Protocol (POP) [2] and Internet Message Access Protocol (IMAP) [3], which were proposed in 1984 and 1996 respectively.

Spam "unsolicited bulk email," is email which the user does not want and it comes without his permission and he cannot easily stop receiving it. The story of the origin of the slang word spam details in Appendix A. Spamming in the electronic communications medium is the action of sending unsolicited commercial messages in bulk without the explicit permission or desire of the recipients [4] . A person engaged in spamming (sending spam) is called a spammer [5].

It is important to note that spam is not only annoying to the individual user, but also represents a security risk and resource drain on the system. It is noteworthy that in developing countries where the bandwidth is limited spam can create unwanted traffic amounting to a kind of denial of service. Email is a cost effective method of marketing legitimate products or services to millions of users, but it can also be used to conduct scams and confidence schemes to steal user information[6].

We assume that in a free multicultural society a spam message is different from one user to another i.e. certain content may be more acceptable to one user but not be acceptable to another. So what is "unwanted" by one user may be liked by another user i.e. what is classified as spam by one user may not be classified by another and by the same user what is classified as spam sometimes with some conditions may not be classified as spam if this

condition changed so the liking of the user changes dynamically. Therefore, there is a need to extend the standard filters to incorporate the interest of the users and the dynamic of opinion change. In this thesis an attempt is made to extend the spam detection to dynamically follow the liking of the user. It is termed personalized spam detection [7].

Most current spam filtering systems have worked in English spam message by relying on the content of the email message there is missing on dynamic English spam detection which would classify spam differently from one user to another. There is a need to have a dynamically user spam detection. It is important to have such detection system for the other languages. People whose mother tongue is a language other than English cannot be forced to use English. They resort to using their mother tongue (e.g. Arabic) or a mixture of English and their mother language (e.g. English and Arabic). This is particularly significant following the new accelerating friendly trend of multilingualism in the internet which is a consequence of the World Summit on Information Society (WSIS)[8]. There are no published detection models on Arabic messages. Currently this situation creates an unjustified disadvantage in the community of Arabic users.

Due to the immense amount of Arabic emails as well as the number of internet Arabic language users, this thesis aims and attempts to provide Arabic spam detection.

1.2 Problem Statement

Spam or unsolicited email is defined by the fact that the recipients did not request the mail or reveal their email addresses for the purposes of receiving such mail. Email has seen explosive growth of usage in the last years, a considerable number of people all around the world use email as a very quick, handy and cheap way of communication. Internet users face many problems in this case. Spam continues to be a growing problem accounting for over 90% of all email today[9]. While spam filters have become more effective and widespread, many spam messages continue to be delivered to end users. This problem has negatively impacted consumers, businesses, and Internet Service Providers (ISPs) because spam represents a security risk and resource drain on the system. It is noteworthy that in developing countries where the bandwidth is limited spam can create unwanted traffic amounting to a kind of denial of service. A possible cause of this problem is that producers most continuously seek new ways for advertising their products in

order to maximize their sales. Therefore, sending emails to people would be a reasonable way of advertising products since it is very cheap, quick and an effective way for organizations and businesses spam handling results in huge financial losses, in addition to being a source of annoyance. A 1000 user organization spends more than $1.8 million to take care of spam, according to the estimates provided in[10].

The usage of the internet throughout the Arab world is witnessing a rapid increase every day. The total of population in Arab countries is around 350 million people (5% of the World population), and the total of Arab internet users is around 65 million users (3.3% of the total internet users) [11]. In recent years, the ratio of Arabic email spam messages has increased a lot and the users of Arabic emails face the problem of spam on a very large scale but the research in this area is not as advanced as its counterpart for English spam.

The basic problem of this thesis is to create a model that can dynamically classify spam (English and Arabic) from legitimate messages.

In this thesis, a dataset of Arabic emails is collected and I used them to train different algorithms in order to classify between spam and non-spam. Naïve Bayesian is used to classify English emails (developed model by the Matrix laboratory (Matlab)) [12] and modified to classify Arabic emails using a Python programming language [13] is used to build an Arabic spam detection model. Cross validation experiments are used to evaluate the model. An Active Server Pages (ASP) [14] programming language is used to develop a web based spam detection system we called it Permail.

1.3 Objectives

The main objectives of this thesis are to:

- Build a dynamic and personalized model to detect English spam emails and then test the model against a standard data set.
- Modify the English model to detect Arabic and mixed (English and Arabic) spam emails.
- Collect and build an Arabic corpus for testing the Arabic spam detection model.
- Create Arabic spam words for using in spam filters.
- Develop Personalized spam detection web based (Permail) and compare the spam filtering capabilities of Microsoft Hotmail, Google Gmail, Yahoo Mail and Permail to determine the effectiveness of

spam filtering for each provider. The key measurements for this thesis are the quantity and percentage of spam in the Inbox.

1.4 Methodology

The methodology of this thesis is to build spam detection framework include English spam detection model, Arabic spam detection model, web based personalized spam classification system, and create Arabic spam corpus.

1.4.1 English spam detection model

1. Use Matrix laboratory (Matlab) [12] to develop a personalized English spam detection model
    - Use emails corpus from Second Conference on Email and Anti-Spam CEAS 2005, Stanford University, Palo Alto, CA [15]
    - Apply Naïve Bayesian (NB) to the proposed model to gain the advantage of the Naïve Bayesian classifier.
    - Perform the preprocess step and eliminated the common words from the message which we are going to classify.
    - Since different users receive different types of legitimate emails, the training process of model probabilities of spaminess of words is computed differently and as a result we would have different classification for each user.
    - On the test phase, we find out the most interesting words of that message. Afterwards, we find out the spam message which contains this word. From Bayes rule.
    - Design graphical user interface (GUI) to check whether the given email is classify as either spam or non-spam email.
    - Evaluate the model by using recall and precision Matrix algorithms[16].

1.4.2 Arabic spam detection model

2. Use Python programming language [13] to build an Arabic spam detection model
    - Modify the Naïve Bayesian model built for English spam emails to create the Arabic spam detection model.
    - Build a dataset of Arabic emails include spam and non-spam. The first part is called a training dataset to use to

build the model, and contains around 700 messages. While the second part is called a test data set, contains around 300 messages, and used to evaluate the model.

- Extract a set of features from the Arabic dataset.
- Perform feature selection.
- Evaluate Naïve Bayesian algorithms to classify incoming email as spam or non-spam.
- Use 10-fold cross validation to calculate the accuracy of the classifiers as the following:
    - Break data into 10 datasets of size n/10, where n is the data size.
    - Train the model on 9 datasets and test it on one dataset.
    - Repeat 10 times and take a mean accuracy[17].

1.4.3 Web based spam detection system

3. Use an ASP programming language to build a spam detection web based system we call it Permail
    - Use MS Access to build an email database.
    - Use Object Database Connectivity (ODBC) [18] to create a connection string.
    - Develop ASP program files that can perform the transactions between the front end (web pages) and back end (database).
    - Examine the performance of personalized spam detection email by creating many users and see the dynamic content of whitelist, blacklist and vocabulary list which is different from one user to another, this showed the dynamic performance of the system and personalized user lists.
    - Use English published corpus to send about 102 emails (spam and non-spam) to Permail, Hotmail, Yahoo and Gmail systems.
    - Compare the classification results of Permail, Hotmail, Yahoo and Gmail to identify the best classifier to detect spam emails.

1.4.4 Creation of Arabic corpus

Collect and build an Arabic corpus for testing the Arabic spam detection model. Create Arabic spam words for using in spam filters.

1.5 Thesis Importance

This thesis tacks his importance from the following main point for the Arabic spam detection, there is no:

- Standard Arabic emails corpus and Arabic spam words use in spam filters.
- Spam detection model for Arabic emails.
- Dynamic spam detection web based email system.
- Published literature for Arabic spam classification.

1.6 Contributions

This section presents the contributions and contains papers, publications and others as follows:

1.6.1 Publications

This thesis has produced three papers

- Asma Ibrahim, Izzeldin Mohamed Osman, "A Behavioral spam Detection System", Advances in Intelligent and Soft Computing, Future Computer, Communication, Control and Automation ABC, ISBN 978-3-642-25537-3, ICEA Conference- Shanghai, China 2011.
- Asma Ibrahim, Izzeldin Mohamed Osman, "Arabic spam detection model", Submitted for publication to World of Computer Science and Information Technology Journal (WCSIT).
- Asma Ibrahim, Izzeldin Mohamed Osman, "Personalized spam Detection model."

1.6.2 Other Contributions

In addition to publishing the papers mentioned in the above paragraph the thesis has the following additional contributions

Build an international Arabic email corpus – An Arabic emails data set was collected to perform training and testing of Arabic detection models.

Arabic spam detection model – A model to detect and classify Arabic emails into spam and legitimate was developed. This model used Naïve Bayesian algorithm to classify the messages.
Personalized Spam Detection Web Based System - A dynamic email system to classify spam message into spam and legitimate was developed. This system depended on the behavior of the user and built dynamically, personalized whitelist, blacklist and vocabulary list.
Create Arabic spam words – This words can use in Arabic spam filters.

1.7 Thesis Organization

This thesis is divided into six chapters: Introduction, Literature review, Arabic email corpus, Arabic spam detection model, personalized spam detection algorithm, and Conclusion.

Chapter 2: Literature Review
This chapter gives a review of (all spam solutions) the well known spam detection methods with their advantages and disadvantages; also presented the machine learning algorithms and discuss the Naïve Bayesian in details.

Chapter 3: Arabic Email Corpus
This chapter gives a short review of English email corpus, presents the importance of corpus in Natural Language Processing (NLP) process; also shows the lack of Arabic email corpus, shows the need of Arabic email corpus and describes the proposed corpus.

Chapter 4: Arabic Spam Detection Model
This chapter discusses in details the Arabic spam detection model, testing the model, the experiments and results.

Chapter 5: Personalized Spam Detection Algorithm
This chapter discusses in detail the architecture of the personalized spam detection model, present all screens of the dynamic email system which is used to detect spam messages dynamically, testing the model, the experiments and results.

Chapter 6: Conclusion
This final chapter contains a summary of the content of the thesis, and recommends areas for future research.
A list of references follows and Appendixes.

CHAPTER 2: LITERATURE REVIEW

CHAPTER TWO

LITERATURE REVIEW

The problem of unsolicited bulk email, or spam, gets worse with every year. The big amount of spam being sent wastes resources on the internet and wastes time for users. This development has stressed the need for spam filters [19], several filters have been built to protect from this problem.
This chapter gives a review of the well known spam solutions with their advantages and disadvantages. There are many studies to filter spam of English email which use the Naïve Bayesian because it is the one of the best spam filter algorithms and NB classifier can outperform other powerful classifiers when the sample size is small [20]. There are many studies of Arabic web pages spam but there is a scarcity of published studies of Arabic email spam and there is a need for personalized spam detection system that produced a new personalized feature not found on the other famous email systems(Gmail, Hotmail or Yahoo).The following literature review attempt to present the previous studies and related works.

2.1 Spam Solutions Approach

Many different spam filtering approaches have been tried in filtering models. Most of these have a degree of effectiveness and drawbacks. The six most significant spam filters are discussed below, along with their strengths and weaknesses.

2.1.1 Rule Based Approach

With the rule based approach, each email is compared with a set of rules to determine whether it is spam or not. A rule set contains rules with various weights given to each rule. Initially, each incoming email message has a score of zero, then, the message is passed to detect the presence of any rule. If any rule is found in the message, its weight is added to the final score of the email. In the end, if the final score is found to be above some threshold value, the email is declared as spam[21]. Rules are observations of features that are found more frequently in spam than in legitimate messages.

Advantages

This approach can be very effective with a given set of rules. It can achieve 90 to 95 percent efficiency. The filter is easy to install, it requires copying the rule set. It requires neither training nor any sort of personal tuning.

Further, the rule set can be updated by copying an additional set of rules to challenge the current trend of spam [22].

Disadvantages

The disadvantage to the rule based approach is there is no self learning facility available for the filter. Spammers with knowledge of the rule set can design a spam to deceive the method. For example, if there is a rule for classifying a message as a spam if the message contains the word "Wine" more than five times, the spammer can easily circumvent the rule by using the term "W*i*n*e" instead of "Wine." Rules cannot be kept secret. The best option is to go through every spam and update the rule set by manually adding new found rules. Unfortunately, this updating process is never ending, as the spammers continually devise new procedures to deceive the spam filters. This process requires personal effort, time, and some level of expertise, qualities not found in every email user [23].

2.1.2 Blacklist Approach

This technique simply involves organizations manually keeping a list of the Internet Protocol (IP) addresses of known spammers (a "black list") so that emails from those addresses are blocked [24].

2.1.3 Whitelist Approach

Whitelists contain legitimate addresses. The email messages arriving from any of these addresses are allowed to pass into the recipient's mailbox. The messages with sources that are not whitelisted are considered to be spam. It is difficult to maintain an exhaustive list of all legitimate addresses. The better option would be to share whitelists among correspondents, friends and relatives. However, this, too, can be an easy route for a spammer to get a big list of legitimate addresses [25].

Table 2.1 Whitelist and Blacklist Advantages [26]

Blacklist Advantages	Whitelist Advantages
Easty to manage	More secure
Easy to install	More accurate
Can download update quickly	Minimizes false positive
	Can be created at various levels within the enterprise
	Easy to customize

Table 2.2 Whitelist and Blacklist Disadvantages [26]

Blacklist Disadvantages	Whitelist Disadvantages
Exponential growth	More time to manage
Many false positives.	Requires additional time install.
Continual updates are required	Reviews all new IP address
Hard to switch to whitelist	

2.1.4 Signature-based Approach

The signature-based approach compares every new incoming email with the known set of spam [27]. The signature-based approach works in this way each character in an email carries weight. So, the summation of all characters would give a final score that is used as the signature of that email. Thus, every new message's signature is compared with that of a spam's signature. If the signatures match, then the new email is classified as spam [28].

Advantages
The signature-based approach rarely generates false positives. It is usually very fast to compute[29].

Disadvantages
These filters are easy to defeat. Since they are backward looking, they take action only after they become aware of a spam. A small change in emails might make the filter useless. Just by adding some random characters to each spam, the signatures of each will be different from the original spam. Thus, all such spam messages will pass for legitimate messages. In addition, these filters can only be used at the Internet Service Provider (ISP) level as first pass filters [30].

2.1.5 Filters Fight Back

The filters fight back approach is the most aggressive among all the approaches adopted for filtering spam. It employs the policy of "attack is the best self-defense." A spam message usually includes Uniform Resource Locator (URLs) for the readers to visit a site. The purpose may be commercial or social. The filters fight back approach works in this way once a message is detected as a spam, these filters send a number of requests to those URL-specified sites. A user can personally configure the number of

requests. If any spam is sent to thousands of users, there is a high possibility that the server hosting that site would receive millions of requests increasing the cost and the bandwidth, effectively shutting down all its services [31].

Advantages
Since spam it has been the reason for the spammer's loss, spammers would hesitate to send spam to unknown users. More recipients of the spam would create more loss to the spammer's web server [32].

Disadvantages
The job prior to fighting back is to detect a spam. Any URL sent to thousands of users mainly indicates a spam. However, at the bottom of every message, there are many advertisements, such as Yahoo, MSN, etc., many of which are legitimate URLs. If the site turns out to be legitimate, negatively affecting the site might involve legal proceedings. To avoid such confusion, auto-retrieval filters should refer to blacklists for servers that are banned. Further, the servers need to be blacklisted by human intervention, thus ensuring that the auto-retrieval filters send requests only to web servers that are blacklisted.
With this approach, there is an easy way out for spammers. They need to include only active unsubscribe links in their messages. In that way, the senders with auto-retrieval filters will be unsubscribed from the program, which is good news. However, the spam is not reduced globally. There is also the possibility that spammers might include their contact information and their image for marketing purposes instead of their URLs. Doing so, would wholly eliminate the danger of auto-retrieval filters. To make this filter more effective, one needs to fine-tune the filter to each user's incoming message. Fine-tuning a filter requires time and expertise, both of which are often hard to come by. Thus, one needs a filter that is adaptive in nature, one that self learns from the given legitimate messages and spam [32].

2.1.6 Content-Based Filters

Content filtering was one of the first types of anti-spam filters to be used. An example of such filter is spam Assassin [33], which works by scanning the textual content of the email against each rule and adds the scores for all matching rules. If the total score of the email exceeds some set threshold score, then the message is considered spam.
The simplest of content-based rules flag emails if a specific string or expression was matched. For example, one type of content filtering rule

would be to match all emails that contained a variation of the word WINE in the body or subject line[34].
Content-based filters can also match keywords or expressions in other parts of the email such as the headers or the base64 encoding of email attachments and embedded images. Email headers are typically used to determine where an email came from and how it was delivered from the initial source to the current destination[35].

Table2.3: A Summary of the comparison of some spam filtering approachs [36]

No	Approach	Good	Bad
1	Complaining to Spammers' ISPs	Raises cost of spamming.	Laborious
2	Mail Server Blacklists	Block spam right at the server.	Incomplete, sometimes irresponsible
3	Signature-based Filtering	Rarely blocks legitimate mail.	Catches only 50-70% of spam.
4	Naïve Bayesian Filtering	Catchs 99% to 99.9% of spam, low false positives.	Has to be trained.
5	Rule based Filtering	The best catch 90-95% of spam, easy to install.	Static rules, relatively high false positives.
6	Challenge-Response Filtering	Stops 99.9% of spam.	Rude, delays or drops legitimate email.

2.2 Naïve Bayesian Researchs

Taninpong and Ngamsuriyaroj [37] proposed a model for an incremental adaptive spam filtering that would improve the classification accuracy and reduce the misclassification rates. Two significant issues where considered (1) adaptation: the model should be adaptable to rapid and constant changes of spam patterns, and (2) performance: the learning process should be fast and does not require lots of memory. This work used Naïve Bayesian classifier based on a single word representation since it had good performance, simplicity and auto-adaptability. It was modeled as an incremental scheme that received a stream of emails, and applied the concept of sliding window to train only the last few emails for testing new incoming messages. Subsequently, the new features of tested messages were added to the existing features so that the model will be adapted to future incoming emails which gave good performance, simplicity and adaptability.

The proposed model was tested on two corpora: Trec05p-1 and Trec06p [15]. The parameters were the window size and the number of features, and the evaluation metrics were the processing time per message and the non-spam and spam misclassification rates.
The results in this study was that the overall accuracy of the Naïve Bayesian is was better than that obtained from the batch off-line training and the processing time is was reduced significantly. Also the experimental results showed that the number of features has little impact whereas the window size has had significant effects on misclassification rates and the processing time.
Chen et al [38] constructed different filters using three types of classification, including Naïve Bayes, SVM, and KNN. They compared the pros and cons between these three types and used some approaches to improve them to get a better spam filter.
The result indicated that the Naïve Bayesian is a good method of spam filtering, and the time costs less on training (about 1-2 seconds). The result also indicated that testing an input message required much time using Naïve Bayesian, but the result is good enough [38].

Hovold [19] presented the results of using a variety of the Naïve Bayesian classifier for spam filtering. The effects of various forms of attribute selection were explored, as were the effects of considering not only single tokens, but rather sequences of tokens, as attributes. An efficient scheme for cost-sensitive classification is also introduced. All experiments were conducted on several publicly available corpora, thereby making a comparison with previously published results possible.
The result in this study has shown that it is possible to achieve very good classification performance using a word-position-based variety of NB. Results also showed the simplicity and low time complexity of the algorithm thus makes NB a good choice for end-user applications also the results indicated that using the word- position- based attribute vectors gave very good results when tested on several publicly available corpora.

Katirai [39] compared the performance of genetic programming and NB classifiers in the spam filtering domain. The training set for this experiment had seven times as many junk messages after removing duplicate spam. The corpus was tokenized into words, applying both stemming and a stop list. The best results showed genetic programming with junk precision of 95.45% and recall of 70% while Na¨ıve Bayes stood at 95.83% precision and 76.67% recall. Katirai also noted that if certain regular punctuation

sequences (such as signatures and boundaries of forwarded messages) are removed, repeated punctuation is an information-rich feature.

Gee [40] evaluated the effectiveness of a classifier incorporating Latent Semantic Indexing ("LSI") to filter spam email, using a corpus used inprevious studies. This study sought to compare the results of using a Naïve Bayesian classifier with the results from using an LSI-inspired classifier. While using LSI leads to precision roughly equal to that of using a Naïve Bayesian approach, the LSI technique has a substantially higher recall and was generally more effective under certain conditions.
The results shown that using LSI as the basis for an email classifier to filter out spam enjoys a very high degree of recall as well as a high degree of precision, no matter if the corpus was treated using a stop list or a lemmatizer. Also the results of an email classification test where both the recall and precision measurements are both very high and fall into acceptable levels.

Brien and Vogel [41] compared NB to the “Chi by degrees of Freedom” approach. The latter is often used in author detection, and is used under suspicion that most spam is sent by a small number of prolific profiteers. They also compared the effectiveness of both classifiers working on words or characters as features. NB with words produced spam precision of 100% and recall of 76.9%, while operating on characters produced 100% in both categories (on a test corpus of less than 70 messages). The Chi approach produced 100% precision and recall when operating on words and 97.5% precision/100% recall working with characters.

Metsis et al [42] discussed and evaluated experimentally in a spam filtering context five different versions of the (NB) classifier. Their investigation included two versions of NB Flexible Bayesian (FB) and the multinomial NB with Boolean attributes. They emulated the situation faced by a new user of a personalized learning-based spam filter, adopting an incremental retraining and evaluation procedure. They used six datasets, which they make publicly available, were created by mixing freely available non-spam and spam messages in different proportions. The mixing procedure emulates the unpredictable fluctuation over time of the non-spam spam ratio in real mailboxes.
The result indicated that the most interesting result of the evaluation was a very good performance of the two NB versions that have been used less in spam filtering, i.e., FB and the multinomial NB with Boolean attributes;

these two versions collectively obtained the best results in the experiments. Taking also into account its lower computational complexity at run time and its smoother trade-off between non-spam and spam recall, they tend to prefer the multinomial NB with Boolean attributes over FB, but further experiments are needed to be confident. The best results in terms of effectiveness were generally achieved by the largest attribute set (3000 attributes), as one might have expected, but the gain was rather insignificant, compared to smaller and computationally cheaper attribute sets.

Deshpande et al [31] examined the effectiveness of statistically-based approaches Naïve Bayesian anti-spam filters, as it is content-based and self learning (adaptive) in nature. Additionally, they designed a derivative filter based on relative numbers of tokens. They trained the filters using a large corpus of legitimate messages and spam and they tested the filter using new incoming personal messages.More specifically they evaluated different threshold values in order to find an optimal anti-spam filter configuration.
The result indicates that the based on cost-sensitive measures, additional safety precautions are needed for a Bayesian anti-spam filter to be put into practice. However, the technique can make a positive contribution as a first pass filter.

Stone [43] explored and examined the effect of several parameters including corpus size, training set size, feature extraction method, and message portions to filter spam email presented by applying the NB algorithm. He compared the results to several published statistical spam filtering approaches.
The result showed that the best classifier variant classifies messages with 100% legitimate recall and 92.9% legitimate precision with room for improvement — comparable to or better than most published results.

Khalid and Osman [44] introduced a simple feature selection algorithm to construct a feature vector on which the classifier will be built. They conducted an experiment on the SpamAssassin public email corpus to measure the performance of the NB classifier built on the feature vector constructed by the introduced algorithm against the feature vector constructed by the Mutual Information algorithm which is widely used in the literature. The effect of the stop list and the phrases-list of the classifier performance were also investigated.
The results of the experiment showed that the introduced algorithm outperforms the Mutual Information algorithm.

Braganza [45] investigated variable in Naïve Bayesian spam filters. The main focus was to create a method for better spam detection by combining the classical Naïve Bayesian filter with a neural network that analyzes various characteristics of the email body.
The results indicated that the analyzed and the method deemed effective in conditions were very strong thresholds must be set or where the training data is not exhaustive.

Androutsopoulos et al [46] were trained the Naïve Bayesian classifier automatically to detect spam messages, they tested this approach on a large collection of personal email messages, which they made publicly available in "encrypted" form contributing towards standard benchmarks. They introduced appropriate cost-sensitive measures, investigating at the same time the effect of attribute set size, training corpus size, lemmatization, and stop lists, issues that had not explored in previous experiments. Finally, the NB filter was compared, in terms of performance, to a filter that uses keyword patterns, and which was part of a widely used email reader.
The results indicate that NB filter outperforms by far the keyword-based filter, even with very small training corpora.

2.2.1 Advantages of the NB classifier

The main advantage of Bayesian classifiers is that they are probabilistic models, robust to noise found in real data. The NB classifier presupposes Independence of the attributes used in classification. However, it was tested on several artificial and real data sets, showing good performances, even when strong attribute dependencies are present. In addition, the NB classifier can outperform other powerful classifiers when the sample size is small[20]. Since it also has advantages in terms of simplicity, learning speed, classification speed, storage space [46]. Therefore, it is considered one of the best spam classifications Algorithm.

2.3 Arabic Spam

Arabic is a very rich language with complex morphology. The Arabic language is different from other Indo-European languages in terms of its syntax, morphology and semantics[47]. The writing system of Arabic has 25 consonants and three long vowels. The Arabic text is written from right to left and the written letters change shapes according to their position in the word.

The Arabic language consists of three types of words; nouns, verbs and particles.
For instance, a word in Arabic consisting of three consonants like (ك ت ب ktb) “to write” can have many interpretations with the presence of diacritics [48] such as has shown in Table 2.3. Actually the diacritics should be written as they are part of the Arabic word but they are not shown in the common popular Arabic.

Table 2.4 Different interpretation of the Arabic word generated from the text كتب (ktb) using diacritics

Arabic word	Transliteration	Part of speech	English meaning
كَتَبَ	Kataba	Verb	Wrote
كُتُب	Kutub	Noun	Books
كُتِبَ	Kutiba	Passive(verb)	Written
كَتَّبَ	Kattaba	Verb	Make someone to write

In comparison to English, Arabic language is recognized to be sparser, meaning that the Arabic words are repeated less than the English words for the same text length. Thus, in this sense, sparseness results in less weight for Arabic terms (features) compared to the English features. Since the difference of weight for the Arabic word is less than that of the English words, it becomes more difficult to differentiate between the different Arabic words, which consequently may negatively affect Arabic text's classifier’s effectiveness [49].

2.3.1 Arabic Web Pages Spam Researches

Wahsheh et al [11] have analyzed the behaviors of the spammers in the content based Arabic web pages, through analyzing the weight of the most ten popular Arabic words used by Arabic users in their queries. Decision Tree was used to evaluate this behavior and it obtained the degree of accuracy which is equal to 90%.
The results showed that the behavior of the spammers in the Arabic web pages can be unique and distinguished in comparison to other languages.

Al-Kabi et al [50] had investigated four different classification algorithms (NB, Decision Tree, SVM and K-NN) to detect Arabic web spam pages, based on content. They used three groups of datasets, with 1%, 15% and 50% spam contents, and were collected using a crawler that was customized for that study. Spam pages were classified manually.

Wahsheh et al [51] extended an Arabic spam dataset previously built by the authors of Arabic content-based spam web pages. The authors applied the Decision Tree classifier (J48) which is shown to be the best classifier to detect content-based Arabic web spam.The Decision Tree algorithm yields an accuracy of 99.521%, and error rate of 0.479%. A content based Arabic web spam detector is also developed, which extracts the content features of web pages, and compares their features with the rule based (graph structure) of Decision Tree. The content-based web spam detector presents a solution to clean the search engines from Arabic spam web pages.
The results indicated that the content-based Arabic web spam detection showed an accuracy of 83%, using a dataset of 2,500 spam web pages.

Jaramh et al [52] collected a corpus of Arabic web pages(spam and non-spam) manually using search engines such as Google, Bing, AltaVista, Maktoob, Ayna. They proposed a set of new features to enhance the classification of Arabic web pages into spam and non-spam under different classification algorithms, namely Decision Tree, Naïve Bayesian, and LogitBoost. They compared their features, which they called Arabic Content Analysis (ACA) features, to state of the art Content Analysis (CA) features for spam detection in the English web.
The result showed that augmenting the CA features with ACA features achieved an increase in detection accuracy of Arabic spam pages compared to CA features alone. When combined, ACA and CA feature correctly identified 5,536 pages of the 5,645 Arabic spam pages that they used for testing with a false positive rate of 1.9% using the Decision Tree classifier. They also identified the top-ranked features using the Gain Ratio method [52].

Wahsheh et al [53]discussed the current spamming techniques, ranking algorithms of web pages, applied three algorithms (K-nearest neighbor, NB and Decision Tree) that detected Arabic spam pages, and a comparison between their different results.
The result showed that the K-nearest neighbor is better than other used algorithms.

Wahsheh et al [54] said "there are some web sites developers act as spammers and try to mislead the search engines by using illegal Search Engine Optimizations (SEO) tips to increase the rank of their web documents, to be more visible at the top 10 Search Engine Results Page SERP". The main goal of this study was to solve the Arabic web spam detection problem. They discussed the relation between the Arabic web spam types. This study is a continuation of a series of Arabic web spam studies conducted by the authors, where this study was dedicated to building the first Arabic content/link web spam detection system. The constructed dataset contains three groups with the following three percentages of spam contents: 2%, 30%, and 40%. These three groups with varying percentages of spam contents were collected through the embedded crawler in the proposed system. This Novel system is capable of extracting the set of content and link features of web pages, in order to build the largest Arabic web spam dataset. The automated classification of spam web pages used based on the features in the benchmark dataset. The proposed system used the rules of Decision Tree; which is considered the best classifier to detect Arabic content/link web spam. The proposed system helps to clean the SERP from all URLs referring to Arabic spam web pages [54].
The result indicated that the study produced accuracy of 90.1% for Arabic content-based, 93.1% for Arabic link-based, and 89% in detecting both Arabic content and link web spam, based on the collected dataset and conducted the analysis.

2.3.2 Information Science Research Institute's Stemmer

A stemming is a technique used to reduce words to their root form, by removing derivational and inflectional affixes. The stemming is widely used in information retrieval tasks. Many researchers demonstrate that stemming improves the performance of information retrieval systems [55]. The Khoja Arabic Stemmer is a fast Arabic stemmer that works by removing the longest prefix and suffix present in the input word and then matching the rest of the word with known verb and noun patterns using a root library. The stemmer attempts to take into account the unavoidable irregularities in the language in order to extract the correct root from words that do not follow the general rules[56].
The Information Science Research Institute's (ISRI) stemmer uses a similar approach to word rooting as the Khoja stemmer, but does not employ a root dictionary for lookup. Additionally, if a word cannot be rooted, the ISRI stemmer normalizes the word and returns a normalized form (for example,

removing certain determinants and end patterns) instead of leaving the word unchanged. [57]. The ISRI Stemmer requires that all tokens have Unicode string types. If the user uses Python IDLE on Arabic Windows he has to decode text first using Arabic 'cp1256' coding.

2.4 Email Services

Electronic mail (E-mail) is an essential communication tool that has been greatly abused by spammers to disseminate unwanted information (messages) and spread malicious contents to Internet users [58]. Users can access any free email service such as Yahoo Mail, Gmail, Hotmail. There are various technical measures that are currently available, and each can play a role in the battle against spam When combined, these measures can provide a “good enough” solution to the spam problem for email users[59].

2.4.1 Google Mail

Gmail's filters allow users to manage the flow of incoming messages, can automatically label, archive, delete, star, or forward mail [60].
When one user clicks a message as spam the system learns to blocking similar messages and it can use the behavior with all Gmail users to block similar future messages. Gmail uses machine learning algorithms to combine hundreds of factors to classify spam, supports multiple authentication systems[61].
Gmail spam filter is made of many factors depending on various actions/events such as Whenever Gmail system detects that the message has malicious links and trying to scam user through a phishing mail, When the message is sent from unknown/unconfirmed sender, When the user has already marked any such similar mail as spam, When the message is in different language, When many people marked same message as spam, When the message is sent lesser known domain, When it has similarity to suspicious messages and proactive words like “Get Rich Quickly” or “Free Goodies”[62].

2.4.1.1 Gmail spam filter technologies

Gmail team used priority Inbox to automatically sort incoming email and help users to focus on the messages that matter most. Also, they launched smart labels, which help users to classify and organize their email[63].
To protect Gmail users from spam images, Gmail used Optical Character Recognition (OCR) developed by the Google Book Search team [64].

Machine learning algorithms developed to merge and rank large sets of Google search results allow to combine hundreds of factors to classify spam. Gmail supports multiple authentication systems, including Sender Policy Framework(SPF), Domain Keys, and Domain Keys Identified Mail (DKIM)[65], so user can be more certain that the mail is from who is says it is from. Also, Gmail puts all senders through the same rigorous checks[66]. Gmail used Completely Automated Public Turing-test to tell Computers and Humans Apart (CAPTCHA). CAPTCHA is introduced in 2000 by Luis von Ahn and his colleagues. A CAPTCHA is a challenge response test used to ensure that the response is generated by humans. CAPTCHA test are administrated by machines to differentiate between humans and machine [67].

2.4.1.2 Gmail spam filter is insufficient

Unrealistic to expect users to configure an individual filter for each domain so the spam keeps coming in after added filters for specific hosts[68]. Some users think that maybe Google give the spammers the email address of the users. Or sell it, because they never use the account but got spam[69].

2.4.2 Microsoft Mail

Early versions of the Windows Live Hotmail spam filter consider all mail that does not come from trusted sources as spam and move it to the junk folder. It is not perfect, so a one of spam will arrive in the Windows Live Hotmail Inbox, but the majority will go to the junk folder automatically. At the same time only few legitimate emails will be filtered out by mistake. It decomposes an email into tokens, normally words, but sometimes other markers and then uses the frequency of each word's appearance in that given email against a know sample to determine if it is spam or not[70].

It is great to keep spam out of the Inbox, but user still needs to visit his junk folder every now and theun to make sure he/she has not missed something important. Hotmail not only keeps more spam out of Inbox, it keeps obvious spam out of the junk folder to make this job easier. In fact, they reduced the size of customers' junk folders by more than 50% - that is about a half a billion fewer pure spam messages per day [71].

Microsoft office produces Outlook 2007 contains a junk email filter designed to reduce the unwanted email presented in user Inbox. The junk email filter evaluates each incoming message to assess whether it may be spam, based on several factors. The Outlook junk email filter does not stop

junk email from being delivered, but rather diverts suspected spam to the user's junk email folder instead of Inbox [72].
Outlook 2013 uses Microsoft word as its rendering engine, HTML and Cascade Style Sheet (CSS). There are five different user-controlled lists that can determine whether the email will hit the junk folder or not:

- Safe Senders List – Email addresses and domain names in this list are never treated as junk,
- Safe Recipients List – Users can add the mailing lists and distribution lists that they are part of to this list,
- Blocked Senders List – Once users add an email address or domain to this list, messages from that source are automatically sent to the junk Email folder,
- Blocked Top-Level Domains List – To block messages from another country or region, users can add country/region codes to this list and
- Blocked Encodings List – To block messages that contain another character set or alphabet, users can add encodings to this list [73].

2.4.2.1 Hotmail spam filter technologies

Early versions of email spam filters were very primitive and the mere presence of a word in an email was enough to push it into the spam queue. Those simple measures have been replaced by the more robust Bayesian spam filtering [71].

2.4.2.2 Hotmail spam filter is insufficient

There is no personalization because some messages come to even junk folder when it's deleted them daily[71].
Another failure in this spam filter is what is called ‘Dr Oz’ emails that are getting through partly because the spammer is taking advantage of a key failure in Outlook’s junk Email filter. The built-in filter that is configurable by the user works by matching the apparent email address of the sender. That is fairly useless because spammers change the ‘From’ email address very regularly. Each of the messages has the same ‘From’ name but the address itself is different for each email – spammers do that because they know Outlook’s mail filters are based around the email address [74].
Hotmail filter use Bayesian which it searches for spammers' words, if the spammers' word does not have a space or recognized word separator on either side of it, it will not be picked up[70]. The spammers are no longer using domain names with words that’d be clearly identified as spam. If the

domain itself is not in a common format the spam classifier is not smart enough to identify that.

2.4.3 Yahoo Mail

Collaborative spam filtering in open membership systems, such as Yahoo Mail, depends on user generated label information. Users provide feedback by labeling emails as spam or not spam. These labels are then used to train a spam filter. Although the majority of users provide very little data, as a collective the amount of training data is very large (many millions of emails per day). Unfortunately, there is a substantial deviation in the users' understanding and liking of what constitutes spam and non-spam. As a result, spam filtering based on a global classifier will be sub-optimal. Conversely, there is often insufficient personal information to train an individual classifier for all users. [75].

In this thesis, we will show that IP address is possible to start initially with the global black list and then can edit this list for each user individually.

2.4.3.1 Yahoo Mail spam filter technologies

In the early Yahoo spam filter, the filter consided all mail that did not come from trusted sources as spam and moved it to the junk folder. Recently Yahoo used many techniques to detect and filter spam such as Spam Guard is a feature that automatically moves the spam email to the spam folder. Spam Guard adapts to the individual preferences every time user marks an email using the "spam" and "Not spam" buttons.
Filters function according to rules that the user sets up because the user can automatically filter incoming email into the folders of his choice., A user can block an email address (or domain), if he/she does not want to get emails from that address in Yahoo Mail, Opting to not display images in his received emails can help fight against spam [76].
Thus, this spam Guard makes a good attempt towards personalizing the filter. However, this is done manually.

2.4.3.2 Yahoo Mail spam filter insufficient

There are many failures in the Yahoo spam filter for example each time user ticks spam message and "report as spam" and get thanked for improving "Spam Guard" but it comes straight through to the Inbox [77]. Spammers can log into Yahoo accounts because Yahoo stores password in clear text and having them stolen by hackers. This has been a real headache,

mine spammers for email addresses, and then sends email from those accounts. Then they have been creating other fake accounts that are similar using those names so even if you change your password there may be a Gmail or Hotmail account owned by the spammer using your contact list and name [78].

2.5 Conclusion

many years ago, established progressive principles for detection and filtering spam for English email and some in spam for Arabic web pages. The above studies, while tracing the English spam and Arabic web spam, do not address the important part of non-English or Arabic spam. More studies in the spam of Arabic email should be done.

A second section of the literature review presents the spam filters of the famous mail systems are used by most email users like Gmail, Hotmail or Outlook and Yahoo and their successor and failure or insufficient in filtering spam from user Inbox.

Taken together, the results indicate that spam filter of the big free mail systems that are used by most mail user worked very hard to fight the spammers and always developed new techniques to stop spam or to reduce false positive from mail users, but as presented above there are many complaints from users and this system still needs much hard work to solve and end the problem completely.

In proposing mail services "Permail" attempted to give new and more sufficient spam filter depend basically on personalized behavior and can filter spam message from Inbox using many factors.

The concept of automate personalizing of detecting and filtering spam is not addressed in the literature and not yet satisfied by products.

CHAPTER 3: ARABIC EMAIL CORPUS

CHAPTER THREE

CHAPTER THREE
ARABIC EMAIL CORPUS

3.1 Introduction

A number of large corpora have been assembled in the last few years, but the idea itself is not new. It can be traced back to the German linguist Kading, who in 1897 used a large corpus of German - 11 million words - to collate frequency distributions of letters and sequences of letters [79].

The importance of corpora to language and linguistics studies is aligned to the importance of empirical data. As language and linguistics studies cannot rely on intuition or small samples of language; they require empirical analysis of large database of texts as in the corpus-based approach.

Modern computers have made it possible to store a large number of texts and to analyze a large number of linguistic features in those texts [80].

3.2 Corpus in Natural Language Processing

Natural Language Processing (NLP) is an area of computational linguistics which is interesting to use corpora in computationally than linguistic [81] the researchers in NLP have their own distinct interests and there are a limitations of the corpora created by researchers in NLP especially in Arabic language.

Examples of corpora in NLP are:

In Lancaster University there is a group of descriptive and computational linguists who worked together not only to create the British National Corpus but also to develop the tagger (CLAWS) that was used to tag the corpus [82].

BulTreeBank project Head-driven Phrase Structure Grammar (HPSG-based Syntactic Treebank of Bulgarian) created a high quality set of syntactic structures of Bulgarian sentences within the framework of HPSG it aims to contain samples of all the syntactic structures of the language. These sentences should serve as templates for future corpora development, become the basis for the development of a more comprehensive test suite for NLP applications they can also be used as a source for grammar extraction and for linguistic research [83].

3.3 Email Corpus

This section gives a brief description of the famous English emails corpus.

3.3.1 English Email

- TREC05 Corpus

  TREC's spam Track introduces a standard testing framework that presents a chronological sequence of email messages, one at a time, to a spam filter for classification. The filter yields a binary judgment (spam or ham [i.e.non-spam]) [84].

- Enron Email Corpus

  This dataset was collected and prepared by the A Cognitive Assistant that learns and Organizes Project (CALO). It contains data from about 150 users, mostly senior management of Enron, organized into folders. The corpus contains a total of about 0.5M messages. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation of Enron [85].

- Enronsent Email Corpus

  The EnronSent corpus is a special preparation of a portion of the Enron Email Dataset designed specifically for use in Corpus Linguistics and language analysis. Divided across 45 plain text files, this corpus contains 2,205,910 lines and 13,810,266 words. This preparation was created by cleaning up a portion of the original Enron Corpus [85].

- TREC07 Corpus

  TREC-7 corpus is made up of 75,419 real emails, comprising 25,220 ham and 50,199 spam emails. All emails were arranged in chronological order to simulate a real spam filtering scenario in which training emails are older than testing ones[86].

3.3.2 Arabic Email

NLP, including information retrieval, Machine Translation and other Natural Language-related disciplines, is showing more interest in the Arabic language in recent years [81]. Suitable resources for Arabic are becoming a vital necessity for the progress of this research. Corpora are an important resource, but Arabic lacks sufficient resources in this field, so a research

project needs to compile a corpus, which represents the state of the Arabic language at the present time and the needs of end users. Many trials have been conducted to build Arabic corpora, but some of them were unsuccessful trials and others were for limited commercial purposes[80].
Due to all the previous discussion about the need for Arabic corpus in general and the absence of an Arabic email corpus, and as there is no Arabic email corpus made to be used in spam classification studies there is an urgent need to fill this gab.

3.4 Arabic Email Corpus

Over the past decade, there has been some important progress in the computational processing of Arabic. However, Arabic is still lacking Arabic email corpus. Arabic NLP is still in its infancy, due to the problem of obtaining large amounts of text data [87]. This section will describe the steps that used to build the first Arabic Email Corpus (AEC).

3.4.1 Goal of AEC

We have planned AEC to contain 2000 emails. The collection of samples is of written Modern Standard Arabic (MSA). Demonstrating those corpora have proven to be very useful resources for linguists who believe that their theories and descriptions of Arabic should be based on real, rather than contrived data.

3.4.2 Design of AEC

We built a corpus of Arabic emails containing 1066 messages, 512 spam and 554 non-spam. We collected them from October 2012 to July 2013. The corpus was collected manually from user emails (e.g., Gmail, Hotmail and Yahoo mail). Then we classified it manually as spam and non-spam.
Non-spam emails, messages collected from my personal email and some colleagues whereas spam email messages collected from email of my supervisor (Hotmail), my husband's email and some of my personal email accounts (Gmail and Hotmail).

3.4.3 Description of AEC

This was collected and prepared by the A Personalized Arabic spam Detection Project (PASDP). It contains data from about 8 users accounts, organized into two folders (spam and non-spam). The corpus contains a total

of about 1.29M messages. This data was originally made public and posted to the Sudan University of Science and Technology website and known as SUST corpus at http:/www.susteh.edu/sustcorpus.

CHAPTER 4: ARABIC SPAM DETECTION MODEL

CHAPTER FOUR
ARABIC SPAM DETECTION MODEL

4.1 Spam Detection Process

Email is a cheap method of sending and receiving messages over electronic communication systems such as the internet[88]. Spam “unsolicited bulk email”, is email which the user does not want and it comes without his permission and he cannot easily stop receiving it[89].

There are many techniques have been implemented for English spam emails. Most spam filters use the Naïve Bayesian [19][31][37-46] recently automated anti-spam filters have become a familiar method in spam detection. Since such filters are quite effective, we believe that this classifier they proposed can be implemented by Arabic spam emails to gain better performance.

Developing text classification systems for Arabic documents in general and Arabic emails is a challenging task due to the complex and rich nature of the Arabic language [90]. The Arabic language consists of 28 letters, and written from right to left and it has complex morphology [91]. Arabic exhibits two genders: masculine and feminine, three number categories: singular, dual, and plural. Whereas singular and plural are familiar categories of most Western learners, the dual is less familiar. The dual in Arabic is used whenever the category of two applies, where it is in nouns, adjectives, pronouns, or verbs. The Arabic plurals are divided into two categories: regular and broken. A noun has three cases, the nominative, accusative, and genitive [92].

There is a large Arabic community of users using email services were safer getting rich of the Arabic spam email problem. They need sufficient and a power Arabic spam filter. In the Arabic spam area many Arabic authors worked on Arabic web spam and had many published work [11][50-54], but in fact Arabic email spam area is very poor no published research [93].

This chapter discusses the spam detection model architecture for detection of Arabic emails and mixed (English to Arabic) emails as shown in Figure 4.1. In addition, it presents the importance of preprocessing for Arabic

emails. The first available spam filtering system built upon the principles of the Naïve Bayesian [94].

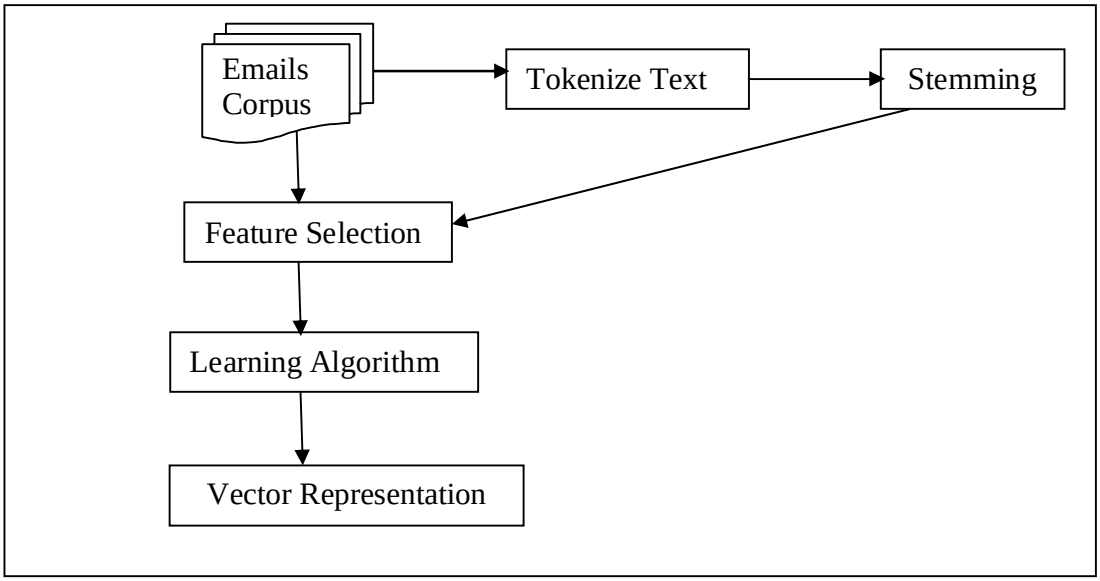


Figure 4.1 Spam Detection Process

The NB method was used to filter email by both Androutsopoulos et al. [46] and Graham [95].
In NB classification, each email is represented by a vector

$$\vec{x} = \langle x_1, x_2, x_3, \dots, x_n \rangle \text{ -----(1)}$$

Where $x_1, \dots, x_n$ Are the values of attributes $X_1, \dots, X_n$, and n is the number of attributes in the corpus of emails that has been collected, each attribute represents a particular word occurring or not. If the email contains the word corresponding to x_i, then $x_i = 1$ otherwise $x_i = 0$. By [96] using Bayes's theorem and the theorem of total probability, that given the vector $\langle x_1, x_2, x_3, \dots, x_n \rangle$ of a document d, and where k $\in$ {spam, non-spam}, the probability that d belongs to category c is as given in (2)[41].

$$P(c|\vec{x}) = \frac{P(c).P(\vec{x}|c)}{\sum_k P(k).P(\vec{x}|k)} \text{ ----(2)}$$

$$P(c|\vec{x}) = \frac{P(c).\prod P\ (x_i|c)}{\sum_k P(c).\prod P\ (x_i|k)} \text{ ---(3)}$$

The probabilities $P(\vec{x}|c)$ (ie. The probability of vector $\langle 1,0,0,1,0,1,\ldots\rangle$ Given C) are almost impossible to calculate, due to the fact that there are too many possible values for $\vec{X}$ Even though it is binary in nature. There are also data sparseness problems [96]. Instead, the NB classifier makes the assumption that X_1,. .. , X_n are conditionally independent of category C. This means that we can change the above equation to the one given in (3). It is much easier to calculate $P(x_i|c)$ Than to calculate $P(\vec{x}|c)$.

For example, it is far less difficult to calculate P ("word"|Category A) than to calculate P (1, 0, 0, 1, 0, 1,. ..) |Category A).
In this instance, $P(X_i|C)$ and P(C) can easily be calculated using the relative frequencies from the training corpus. This is a computationally efficient classifier.

The NB filter calculates the likelihoods of all the words in the given email[97]. The probability of the mail being spam is estimated using equation (3). An effective way to combat false positives is to regard as spam only mails that have a probability higher than a named threshold [41].

4.2 Model Datasets

As any classifier on text documents first we need an Arabic email collection as presented in chapter three we built a corpus of Arabic emails.
English corpus which contains 1200 samples divided into two categories (spam and non-spam) This English corpus is provided by Stanford university [15].

Table 4.1 Summary of Arabic and English datasets

English dataset (Standford University)		
	Number of messages	Percentage
Spam	400	33.3%
Legitimate	800	66.7%
Total	1200	
Arabic dataset		
	Number of messages	Percentage
Spam	100	33.3%
Legitimate	200	66.7%
Total	300	

There are not corpus published for Arabic emails we collect this dataset for this study. Legitimate messages can easily be misclassified as spam. This makes the situation more challenging, as the cost of false positives is much higher than that of false negatives. We feel that by minimizing the false positives in such a situation, we have achieved an efficient Bayesian spam filter. Moreover, by recording tokens from such a huge number of spam, we have covered almost all the topics for spam and are in a pretty good position to classify new incoming mails.
This thesis creates an Arabic spam keywords list contain about 150 keywords which used in Arabic spam filters as shown in appendix B.2.

4.3 Arabic Spam Detection Model

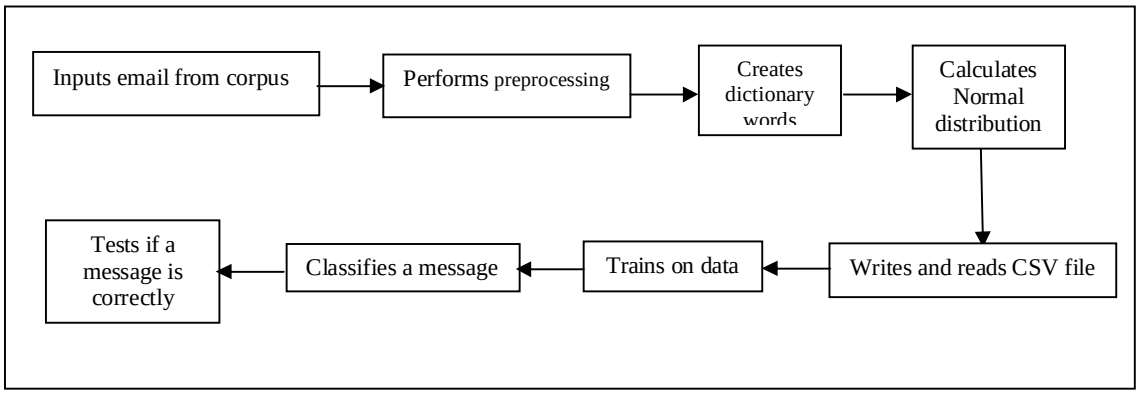


Figure 4.2 Architecture of Arabic spam detection model

We proposed Arabic Spam Detection Model (ASDM) in short, to build the detecting Arabic spam emails. We first perform data preprocessing in the data set, calculate the Document Frequency (DF) given a list of words, we count the number of unique words in the corpus, build a Comma Separated Value (CSV) file to use in the classification process, perform 10-fold stratified cross validation.

4.3.1 Preprocessing

Preprocessing is a very important step before the categorization documents to get knowledge from massive data and reduce the processing operations. [98]. A huge number of features or keywords in the documents lead to a poor performance in accuracy and time [99]. The preprocessing is the first step in data processing of Arabic emails is. It consist of

- Tokenizing

Split each email into words.

- Removal of stop words create a list of stop words such as (etc … ،من، في،الى،التي ،الذين ،هذا ،هذه،إن أن)
- Removal of punctuation:
  Create a list contains all punctuation marks (such as (!), (?), (,), (:) … etc.) and then remove anyone from email if it appears.
- Stemming
  We use ISRIStemmer (open source package) [57] written in python script and then we build method to take a single argument, word, which should either be a cp1256 encoded string, or a Unicode object.
  Which should either be a cp1256 (windows codepage number 1256 which encodes languages which use the Arabic script) string, or a Unicode object.
  The result is the stemmed form of the word. If the word supplied is a Unicode object, the result will be a Unicode object. If the word supplied is a string, the result will be a cp1256 encoded string.
- The normalization
  This step includes converting different form of letters to standardize form such as (أ، آ، إ) to (ا) and changing (ي) to (ى) and finally converting (ة) to (ه). This step aims to unify words typed differently.
- Numeric filter
  Filter all numbers from strings (such as (1), (0.2), (100)… etc)

4.3.2 Document frequency

After the preprocessing of the data we calculate the document (here message) frequency (DF) score for all words in the corpus as sample data shown in the example in Table 4.2. Count number of words in the body and subject, return a dictionary associates each word with the number of times it occurs as shown in the example in Table 4.3. Then count the number of unique words in the corpus.

Table 4.2 Occurrence of words on messages

Word / Msg	تجارة	أخبار	تسوق	بيع	مجاني	العاب	تحميل	أغاني	دردشة	تخفيضات
1	4	0	2	1	2	0	0	0	0	5
2	0	4	0	0	4	0	3	6	0	0
3	0	3	6	0	0	0	0	0	2	3
4	2	0	0	2	0	4	2	0	0	0
5	0	7	0	0	2	6	4	1	0	0

DF Example

In this example, there is a list of five messages. The number of occurrences of each word in all the messages calculates follows:
"تجارة" occurs 4 times in message 1
"تجارة" occurs 2 times in message 4
"تحميل" occurs 3 times in message 2
"تسوق" occurs 6 times in message 3

Table 4.3 Words dictionary of the message

Word	Frequency
تجارة	6
اخبار	14
تسوق	8
بيع	3
مجاني	8
العاب	10
تحميل	9
اغاني	7
دردشة	2
تخفيضات	8

In Table 4.3 for each word on the Table 4.2 the frequency is calculated.

4.3.3 Term frequency

A collection of (n) emails can be represented in the vector space model by a term-email Matrix which it represent for each message the frequency of the words that it contains as shown in the Table 4.2 An entry in the Matrix

corresponds to the “weight” of a term in the email; zero means the term has no significance in the email or it simply does not exist in the email [100].
The most useful terms are those that are of intermediate frequency so most of their occurrences are in a small number of documents in the collection [101].

4.3.4 Cosine Normalization (TFxIDF weights)

Now we can use document frequencies and term frequency to get the term frequency- inverse document frequency (tf-idf) score for every feature (word) of the corpus’s message [102].

$$tfidf(w) = tf.\log(\frac{N}{df(w)}) \quad \text{.....(4)}$$

N: total number of document in a collection.
df(w) : document frequency for feature w (the number of documents that contain w).
idf(w) : inverse of the document frequency.

As shown in the Table 4.4 calculation of document frequency using some words from Table 4.3
N= corpus size = 5

Table 4.4 Example of the document frequency

Word	Tf	IDF
تحميل	9	Log(5/9)
تجارة	6	Log(5/6)
تسوق	8	Log(5/8)
بيع	3	Log(5/3)
العاب	10	Log(5/10)

Calculate weighted term frequency with IDF as show below

$$tf-idf_{td} = tf_{td} \times idf_t \text{ ---(5)}$$

Where
tf_{td}: term frequency for term *t* in document *d*
idf_t: inverse document frequency of term *t* in the corpus.

Table 4.5 Calculation of $tf \times idf$

msg	$tf_{(1)}$	$tf_{(2)}$	$tf_{(3)}$	$idf_{(1)}$ $\log\left(\frac{N}{n1}\right)$	$idf_{(2)}$ $\log\left(\frac{N}{n2}\right)$	$idf_{(3)}$ $\log\left(\frac{N}{n3}\right)$	$w_{(1)}$	$w_{(2)}$	$w_{(3)}$
1	4	0	2	0.398	0.222	0.398	1.592	0	0.786
2	0	4	0	0.398	0.222	0.398	0	0.888	0
3	0	3	6	0.398	0.222	0.398	0	0.444	2.388
4	2	0	0	0.398	0.222	0.398	0.798	0	0
5	0	7	0	0.398	0.222	0.398	0	1.554	0

Normalize the term weight (so longer vectors are not unfairly given more weight). forces all values to fall within a certain range usually between 0 and 1.

$$w_{ik} = \frac{tf_{ik} \log\left(N/df_k\right)}{\sqrt{\sum_{k=1}^{t}(tf_{ik})^2 \left[\log\left(N/df_k\right)\right]^2}} \text{---(6)}$$

Table 4.6 Calculations of Normalized tf x idf Example

	A	B	C	D	E	F	G	H
	$tf(1)$^2	$tf(2)$^2	$tf(3)$^2	$A \times idf$^2	$B \times idf$^2	$C \times idf$^2	D+F+E	$G\left(\frac{1}{2}\right)$
Msg1	16	0	4	2.534	0	0.634	3.168	1.584
Msg2	0	16	0	0	0.789	0	0.789	0.3945
Msg3	0	9	36	0	0.443	5.702	6.146	3.073
Msg4	4	0	0	0.634	0	0	0.634	.317
Msg5	0	49	0	0	2.414	0	2.414	1.207

After calculating (tf x idf) we will organize the data in the tablet format so we will create a CSV file with201 columns - 200 features and class identifier, 301 rows - header (f1, f2...f200, class) and 1066 examples. All sections discussed above have been developed and the source code of all programs on in the Appendix C.3.

4.3.5 Classification Based on Naïve Bayesian

Definition of message classification
Input:

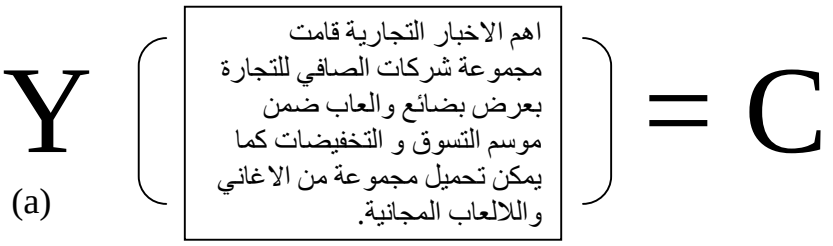
A message m
A fixed set of classes C = {non-spam, spam}

Output:

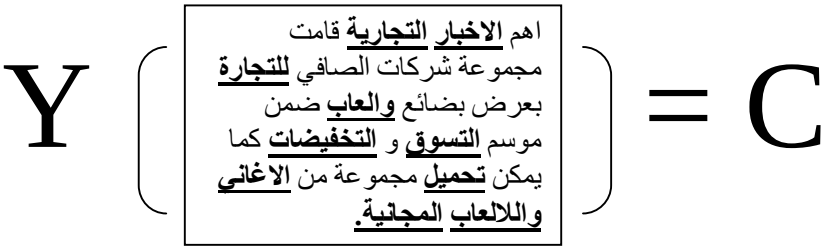
A predicted class

The Naïve Bayesian classification method based on Bayes rule which relies on representation of a document is a bag of words as shown in Figure 4.3 [103]. Bayes' Formula used in email message being filtered to classify the email messages [104].

As mentation before NB represent data in a bag of words here in Figure 4.3 section (a) we shown short messages which we need to classify.



(a)



(b)

In Figure 4.3 section (b) we selected some words that will classify the message if spam or non-spam.

(c)

Figure 4.3 Bag of words representation

In Figure 4.3 section (c) we counted the number of occurrences for each word that selected in (b).

4.3.5.1 CSV files

A CSV is an organized file format use in the text information processing where categorization can be done at word level [105]. On a CSV file each line is interpreted as a text document (here it is email) [106]. We created a CSV file consisted of columns (features and class identifier) rows (header (f1, f2...fn, class) and examples of emails).

First we found the number of examples in the corpus and then calculate tf-idf scores of all messages in the corpus to get the weight of the features then we used this data to create a CSV file consisting of columns.

```
Create headers
 For each index feature in the corpus
 Create the list of [f1, f2,..., fn]
 Append spam class
 Create the list [f1, f2..., fn, spam Class]
Create CSV file
 Append headers to the CSV file
For each message in the corpus
 Append the row of tf-idf scores for each feature
 Append the spam class in the last column
 Append the row in the CSV file
Write the CSV file
 For each row in CSV file
 Write CSV file
```

Figure 4.4 Creation CSV file algorithm

The result of organized data file (CSV) is presented in the Figure 4.5 for body (which contained the data in the body) and Figure 4.6 for the subject (which contained the subject data of the each message) with features (f1..fn) and term class.

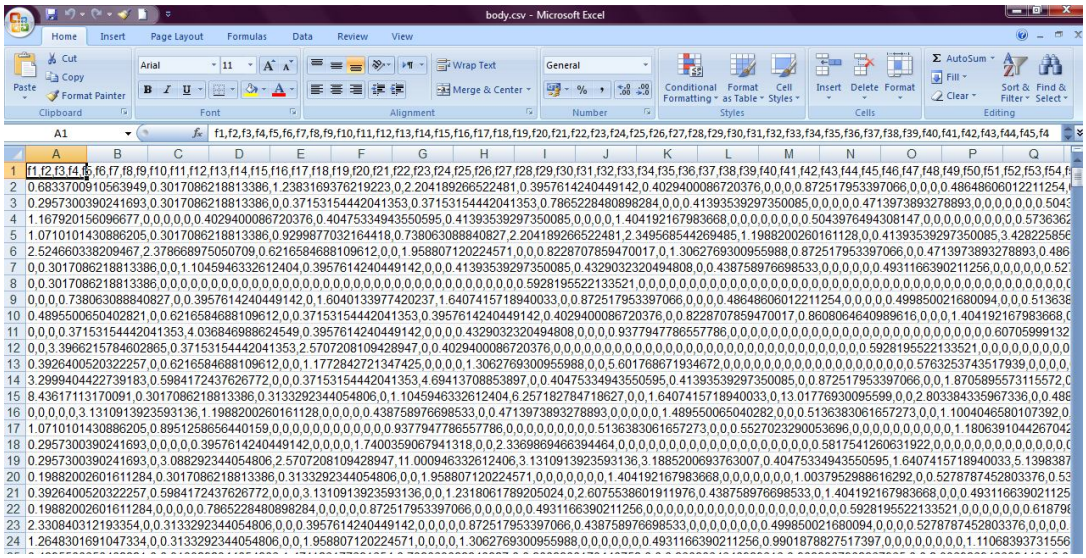


Figure 4.5 CSV file for message body

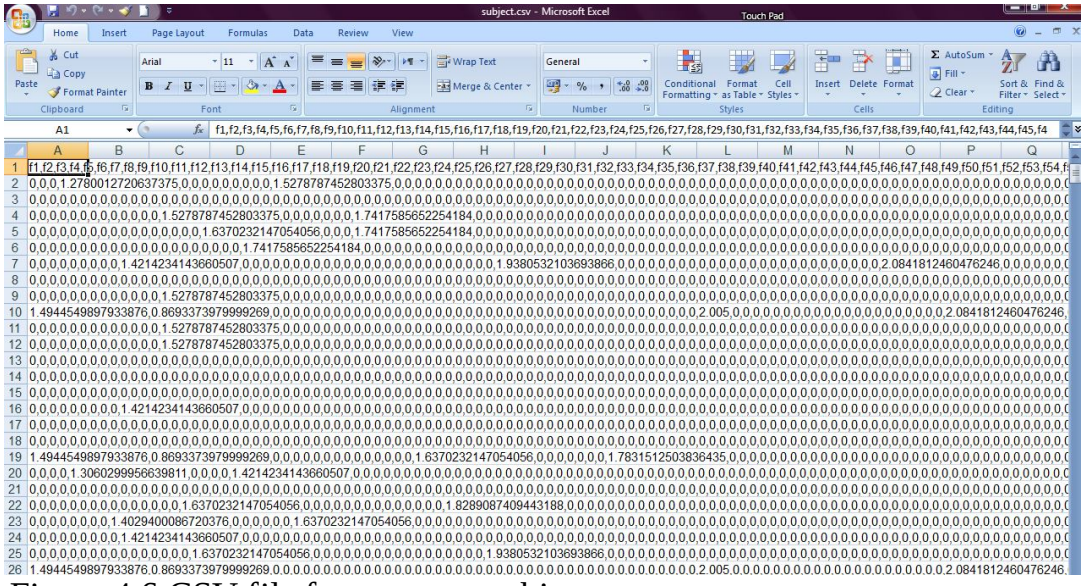


Figure 4.6 CSV file for message subject

In the first step, the classification process retrieved data from CSV, Outputs will be two lists: a list of messages and a list of headers.

```
Read CSV file
Read from the body and a subject CSV file into the reader
For row in reader
Scans through the rows (header & message), appending to the file (corpus data)
Header data "f1, f2...
Message data with TF-IDF scores
For row in corpus data
Converts strings to floats
Return corpus header, corpus float data
```

Figure 4.7 "Read CSV" files algorithm

4.3.5.2 Training

In the training phase, the model is trained using a known corpus of spam and non-spam emails. It keeps track of each word that occurs only in spam, only in non-spam messages, and in both. Based on these word occurrence statistics, incoming unseen messages are processed and classified accordingly. We trained the classifier by calculated the prior normal distribution parameters for the feature sets and true/false and calculates the mean and standard deviation for each feature in training messages when class equal spam class. At the end calculate the priori spam and not spam probabilities.

4.3.5.3 Message classification

To find the classification of the message we calculate the probability that a message is or is not spam, we commenced Bayes probability on message and get a priori class probability of the message. Used feature selection method from WEKA [107] to get feature reduction, finally used tf-idf for each feature of the message to find the probability of the feature and then multiply together to obtain the probability that a message is spam or not spam.

4.3.5.4 Test classification

Once the model is trained using a dataset of spam and non-spam messages, it is ready to perform its basic functionality of classifying new incoming unseen messages.

We used standard deviation technique which emphasized the spam probability of tokens rather than the number of tokens (words). The specialty of the technique is that it assigns the score to the email independent of its size. The same token should not be considered more than once to avoid any interference from the specific token if it had occurred a few times in the message. The processing time for classification would vary according to the size of the email[31].

4.4 ASDM Examples

In this section we presented two examples for spam classification based on ASDM model.

4.4.1 Example (1)

This example uses spam message as shown in Figure 4.8 from Arabic emails corpus, which it uses in ASDM. To classify the email message as spam or non-spam ASDM looks at the subject and the body of the message and apply Bayes rule formula.

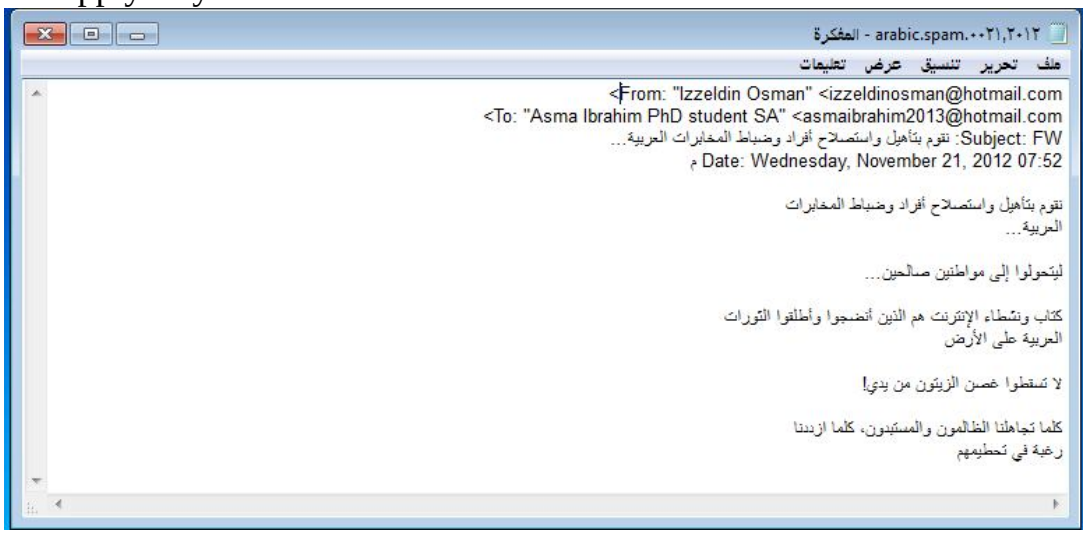


Figure 4.8: Arabic spam message sample

At the first ASDM must split the message into tokens and build a table of all the tokens, the table would be as shown in Table 4.7 then we remove all the stop words as shown in the Table 4.8 finally we perform preprocessing step as shown in the Table 4.9:

Tale 4.7: Tokens of Spam Message

المخابرات	بتأهيل	نقوم	استصلاح	أفراد	وضباط
ليتحولوا	الى	مواطنين	صالحين	كتاب	نشطاء
الانترنت	غصن	هم	الذين	انضجوا	اطلقوا
الثورات	العربية	يدي	على	الارض	تسقطوا
الزيتون	من	لا	كلما	تجاهلنا	الظالمون
كلما	ازددنا	رغبة	في	تحطيمهم	والمستبدون

Tale 4.8: Tokens without stop words

الثورات	العربية	نقوم	استصلاح	والمستبدون	وضباط
الزيتون	ازددنا	مواطنين	صالحين	كتاب	نشطاء
	المخابرات	رغبة	يدي	انضجوا	اطلقوا
	ليتحولوا	بتأهيل	أفراد	الارض	تسقطوا
	الانترنت	غصن	تحطيمهم	تجاهلنا	الظالمون

Tale 4.9: Tokens with preprocessing step

ثورة	عرب	قم	اصلح	استبد	ضبط
زيتون	زد	مواطن	صلح	كتب	نشط
	خبر	رغبة	يدي	نضج	اطلق
	حول	أهل	فرد	ارض	سقط
	نت	غصن	حطم	جهل	ظلم

Once the ASDM has the list of tokens in the message, it searches for the spam and non-spam from the token file.
The file of tokens is created and updated whenever the ASDM is “trained” on a new message.
If a token from the message is found in the file, the ASDM calculates the token’s spamicity based on the following variables:

- The frequency of the token in spam messages that the model has been trained on.
- The frequency of the token in non-spam messages that the model has been trained on.
- The number of spam messages that the model has been trained on.
- The number of non-spam messages that the model has been trained on.

The token’s spamicity is calculated from these pieces of data as follows:

Non-spam probability =
Token frequency in non-spam messages / Number of non-spam messages trained on

Spam probability =
Token frequency of spam messages / Number of spam messages trained on

If either non-spam probability or spam probability are greater than 1.0, we set them equal to 1.0.

Spamicity = Spam probability / (Non-spam probability + Spam probability)

The ASDM was trained on 100 spam messages and 200 non-spam messages. In this example, if we use the token "مخابرات" from the non-spam words of the message, the value will be:

Non-spam probability = 120/200
Spam probability = 0/100
Spamicity = 1.2

This tells us that there's only a 1.2 chance that a message containing the word "مخابرات" is a spam message.

Repeating this process for each of the tokens in our sample message, we get the following frequencies and spamicities:
Table 4.10: Tokens dictionary of the message

Token	Spam Frequency	Non-Spam Frequency
ضبط	322	63
نشط	236	30
اطلق	120	11
سقط	560	10
ظلم	220	22
استبد	145	10
كتب	120	140
نضج	790	29
ارض	230	51
جهل	590	10
اصلح	240	43
صلح	90	29

يدي	470	18
فرد	350	20
حطم	530	29
قم	140	15
مواطن	240	45
رغبة	140	30
أهل	240	98
غصن	130	50
ثورة	430	44
عرب	430	29
زد	140	30
خبر	120	0
حول	340	20
زيتون	140	78

Table 4.11: Token probability and Spamicity of the message

Token	Spam probability	Non-Spam probability	Spamicity
ضبط	3.22	0.27	3.49
نشط	2.36	0.4	2.76
اطلق	1.2	0.6	1.8
سقط	5.6	0.4	6
ظلم	2.2	1.1	3.3
اسبد	1.45	0.2	1.65
كتب	1.2	0.06	1.26
نضج	7.9	0.395	8.295
ارض	2.3	0.2	2.5
جهل	5.9	0.295	6.195
اصلح	2.4	0.445	2.845
صلح	9	0.45	9.45
يدي	4.7	0.15	4.85
فرد	3.5	0.4	3.9
حطم	5.3	0.415	5.715
نقم	1.4	0.1	1.5
مواطن	2.4	0.15	2.55

رغبة	1.4	0.7	2.1
أهل	2.4	0.165	2.565
غصن	1.3	0.065	1.365
عرب	4.3	0.335	4.635
زد	1.4	0.07	1.47
خبر	1.2	0	1.2
حول	3.4	0.1	3.5
ثورة	5.6	0.2	5.8
زيتون	1.9	0.15	2.05

From Table 4.11 the model has calculated the spamicity value for each token in the message.

This message has probability of 92.754 % chance that means the message is spam. If this message was sent to an email server protected by ASDM, it would be tagged as spam.

4.4.2 Example (2)

In this example we use a set of email messages as shown in Table 4.12 for training and testing.

Table 4.12: Spam Messages Collection

	Message	Words	Class
Training	msg1	"صحة."	non-spam
Training	msg2	"صحة الاطفال."	non-spam
Training	msg3	"العاب مجانية."	Spam
Training	msg4	"تحميل العاب."	Spam
Training	msg5	"تحميل العاب مجانية"	Spam
Test	msg6	"تحميل..العاب الاطفال"	?

The first step is to split all the messages into tokens and build a table as shown in Table 4.13 of all the tokens.

Table 4.13: Spam Message Token Frequency

Word	Frequency
"صحة"	2
"طفل"	1
"تحميل"	2
"لعب"	3
"مجاني"	2

Naïve Bayes Learning begins by calculating the prior probability

$$P(c_i) = \frac{count(c_i)}{\sum_j count(j)} \text{----(7)}$$

Applying prior probability equation of spam and non-spam data

$$P(spam) = \frac{3}{5}$$

$$P(non-spam) = \frac{2}{5}$$

Then we have to calculate the conditional probabilities

$$P(w|c) = \frac{count(w,c)+1}{count(c)+|V|} \text{----(8)}$$

Table 4.14: Spam Message Token Frequency and Spamicity

Word	Conditional probabilities	Result
1	$P('تحميل'\|non-spam)$	$\frac{0+1}{5+5} = 0.1$
2	$P('لعب'\|non-spam)$	$\frac{0+1}{5+5} = 0.1$
3	$P('طفل'\|non-spam)$	$\frac{1+1}{5+5} = 0.2$
4	$P('تحميل'\|spam)$	$\frac{2+1}{7+5} = 0.25$
5	$P('لعب'\|spam)$	$\frac{3+1}{7+5} = 0.33$
6	$P('طفل'\|spam)$	$\frac{0+1}{7+5} = 0.083$

The last step is to choose a class for the test message

$$P(non-spam|msg6) = \frac{2}{5} \times \frac{1}{10} \times \frac{1}{10} \times \frac{2}{10} = 0.0008$$

$P(spam|msg6) = \frac{3}{5} \times \frac{3}{12} \times \frac{4}{12} \times \frac{1}{12}$ =0.0041

This test message has probability of 0.0041 chance that means the message is spam.

4.5 Mixed Spam Detection Model

To evaluate the spam detection model that is presented above in mixed dataset English and Arabic spam messages first we need to mix the Arabic and English emails after they classified manually.

Second step we perform data preprocessing separately for each type of message (English and Arabic).

We modify the English spam detection model (from the previous section) to work on mixed datasets.

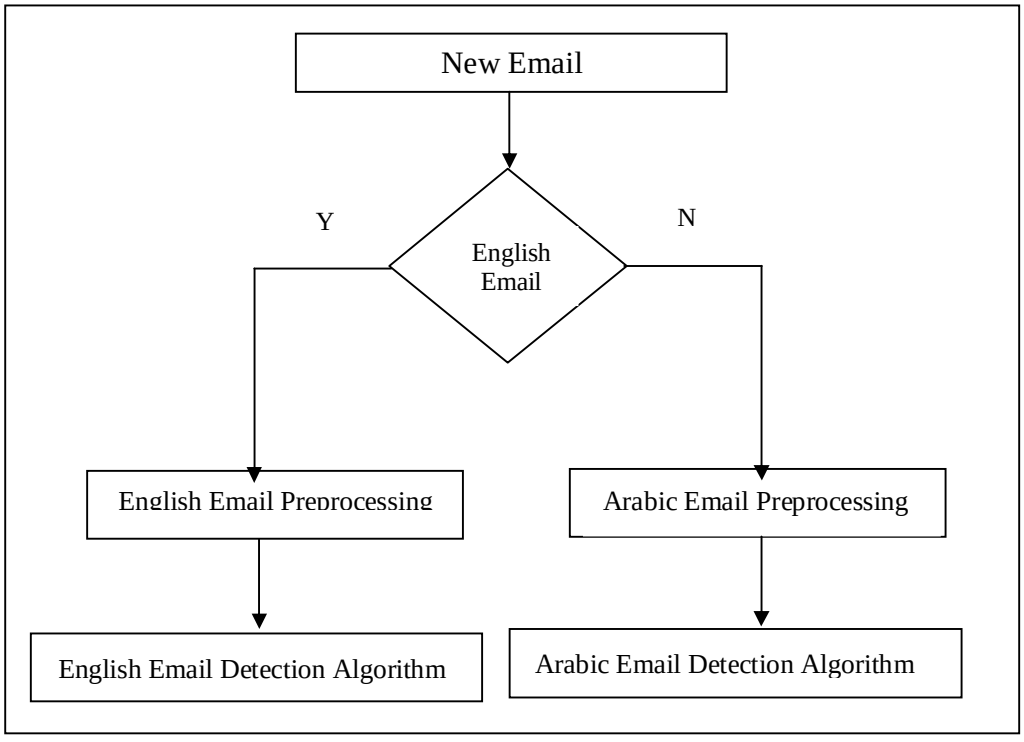


Figure 4.9 Work flow of the mixed spam detection model

4.6 Result and Experiments of the Model

This section explains the set of experiments carried out to evaluate the Arabic spam detection model that we used to classify Arabic spam messages.

In all experiments, 10-fold cross validation was employed. The dataset was partitioned randomly into ten parts, each experiment was repeated ten times, each time reserving a different part for testing, and using the remaining nine parts for training [108]. The final results are the average of the ten iterations. This process produces more reliable results and uses the entire corpus for both training and testing phases [93].

In proposed model we created the set of 10 thirty element random bins after that created the training set (270 elements) and the validation data (30 elements) then train the probabilities of the Bayes filter and calculate the percentage of successful classifications.

4.6.1 Experiment (1) Comparison of Three Datasets:

The first experiment was designed to compare the effectiveness of using the Naïve Bayesian in English, Arabic and mixed in two sections body and subject of the message. The datasets used in these experiments are preprocessed. Table 4.15 and Table 4.16 have shown the results.

Table 4.15 Results of spam detection model for three datasets (English, Arabic and mixed) on Body of the Email

Dataset	Correct Classification	Incorrect Classification	Accuracy	Error
English	577	23	96.2%	3.8%
Arabic	287	13	95.8%	4.2%
English & Arabic	481	19	96.2%	3.8%

Table 4.16. Results of spam detection model for three datasets (English, Arabic and mixed) on Subject of the Email

Dataset	Correct Classification	Incorrect Classification	Accuracy	Error
English	549	51	91.5%	8.5%
Arabic	284	16	94.7%	5.3%
English and Arabic	451	49	90.2%	9.8%

4.6.2 Experiment (2) Applying To The Arabic Mode

The second experiment was designed to get the result when the Arabic spam detection model applied to the body and the subject of the email Table 4.17 and Table 4.18 shown the results.

Table 4.17 Results of Arabic model on the body of email

Stratification set	1	2	3	4	5	6	7	8	9	10
Percentage of Correct classification	94.7%	94.7%	89.5%	100%	100%	89.5%	89.5%	100%	100%	100%
Percentage of Incorrect classification	5.3%	5.3%	10.5%	5.3%	0	10.5%	10.5%	0	0	0
Overall accuracy	95.8%									

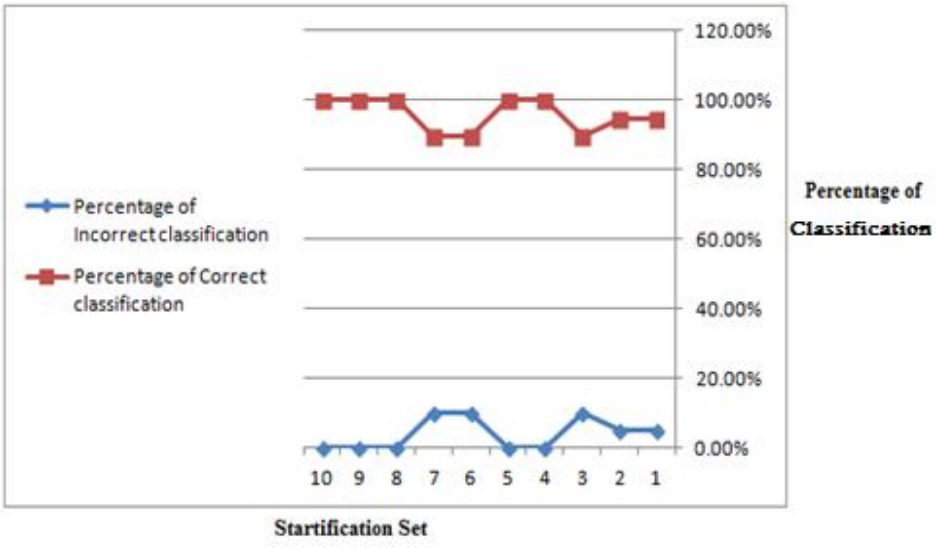


Figure 4.10 Percentage classification of Arabic model of the body

Table 4.18. Percentage of Arabic model on the subject of Email

Stratification set	1	2	3	4	5	6	7	8	9	10
Percentage of Correct classification	100%	94.7%	100%	94.7%	89.5%	84.2%	94.7%	94.7%	100%	94.7%
Percentage of Incorrect classification	0	5.3%	0	5.3%	10.5%	15.8%	5.3%	5.3%	0	5.3%
Overall accuracy	94.7%									

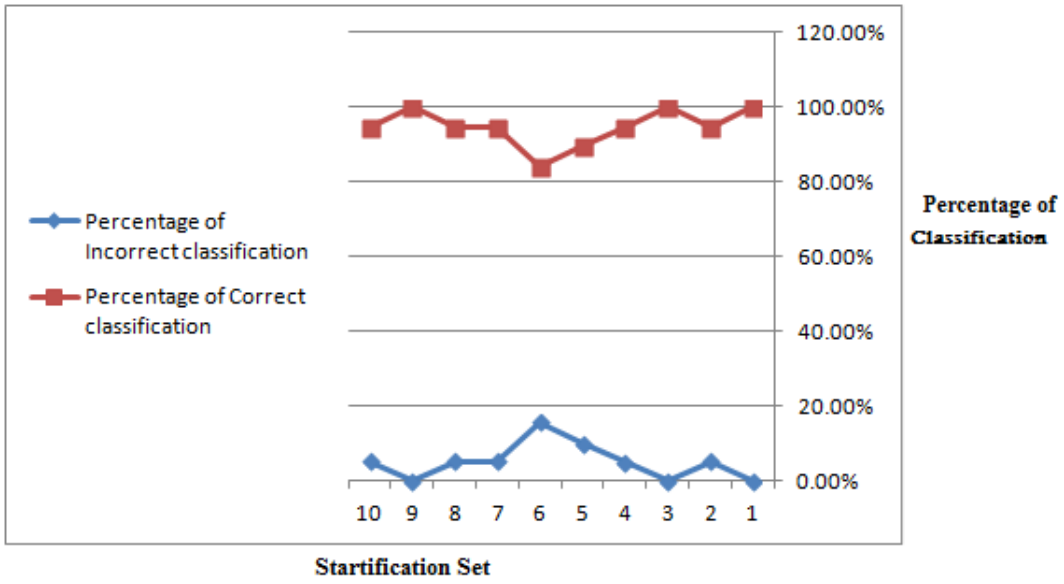


Figure 4.11 Percentage of Arabic model on the subject of email

4.6.3 Experiment (3) Applying On English Model

The third experiment was designed to get the result when applied English spam detection model on the body and the subject of the email Table 4.19 and Table 4.20 shown the results.

Table 4.19 Results of English model of the body of email

Stratification set	1	2	3	4	5	6	7	8	9	10
Percentage of Correct classification	96.7%	100%	95%	93.3%	95%	95%	98.3%	98.3%	93.3%	96.7%
Percentage of Incorrect classification	3.3%	0%	5%	6.7%	5%	5%	1.7%	1.7%	6.7%	3.3%
Overall accuracy	96.2%									

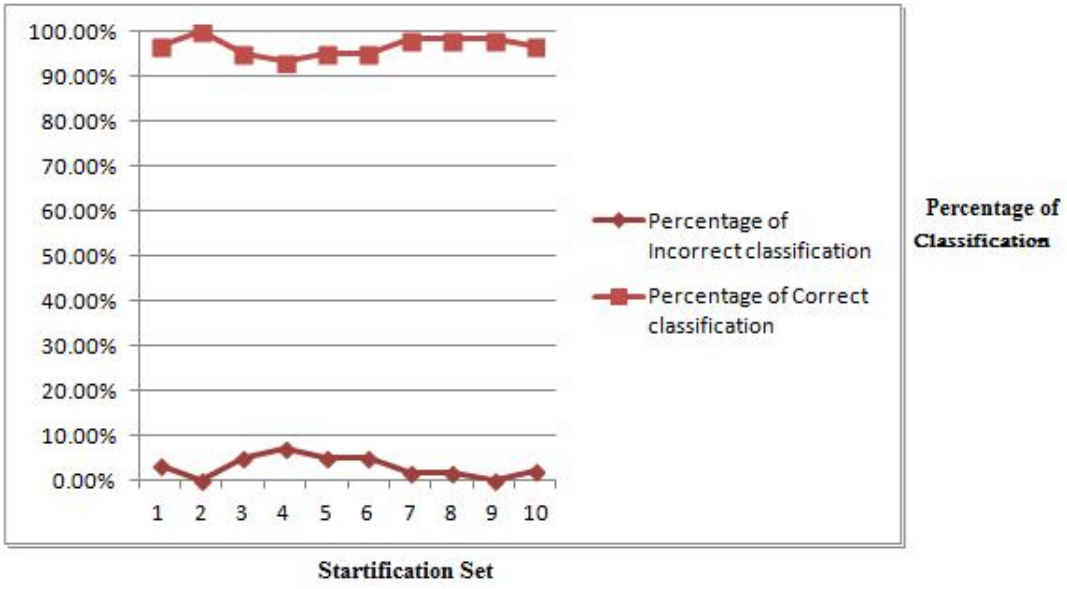


Figure 4.12 Percentage of English model of the body of email

Table 4.20 Percentage of English model on the subject of email

Stratification sets	1	2	3	4	5	6	7	8	9	10
Percentage of Correct classification	100%	94.7%	89.5%	100%	98%	89.5%	89.2%	94.7%	100%	98%
Percentage of Incorrect classification	0	5.3%	10.5%	5.3%	2%	10.5%	10.8%	5.3%	0	2%
Overall accuracy	91.5%									

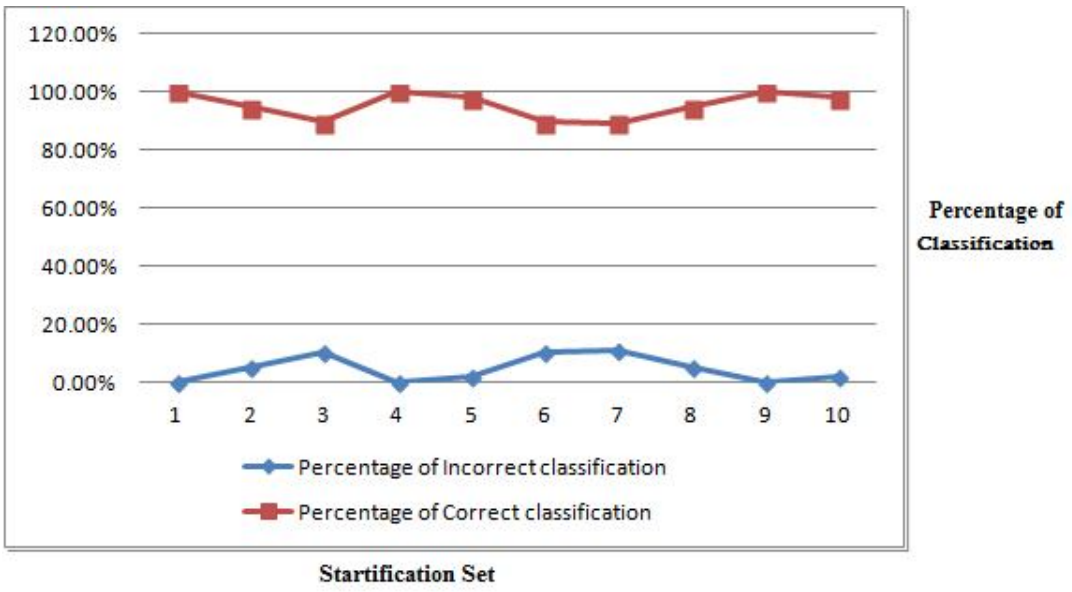


Figure 4.13 Percentage of the English model on the subject of email

4.6.4 Discussion Results or Arabic and English Spam Detection

Accuracy the ratio of correctly classified messages is used as a combined measure. All experiments were conducted using 10-fold cross validation. The reported figures are the means of the values from the ten iterations [19].

The result of model showed that the Naïve Bayesian algorithm is the most effective way to use in detecting Arabic and English spam emails. It is shown that main point effect on the Arabic spam detection is the collection of Arabic corpus and the limit of the size of Arabic dataset.

The results showed that the applied of English and Arabic detection models on the body of the message gave an accuracy of 96.2% of the English messages and 95.8% of the Arabic messages whereas the subject gave less of this accuracy.

4.6.5 Experiments in Mixed Spam Detection Model

This section explains the set of experiments carried out to evaluate the mixed (English & Arabic) spam detection model discussed above that we used to classify spam messages.

In all experiments, 10-fold cross-validation was employed. The final result is the average of the ten iterations. This process produces more reliable results and used the entire corpus for both training and testing phases [93].

4.6.5.1 Experiment (1) applying to the body of the email

The first experiment was designed to get the result when applied mixed spam detection model on the subject of the email Table 4.21 shown the results.

Table 4.21 Results of mixed model on the subject of Email

Stratification sets	1	2	3	4	5	6	7	8	9	10
Percentage of Correct classification	89.2%	100%	89.5%	100%	97%	89.2%	94.7%	94.7%	100%	98%
Percentage of Incorrect classification	10.8%	0	10.5%	0	3%	10.8%	5.3%	5.3%	0	2%
Overall accuracy	90.2%									

4.6.5.2 Experiment (2) applying on the subject of the email

The second experiment was designed to get the result when applied mixed spam detection model on the body of the email Table 4.22 shown the results.

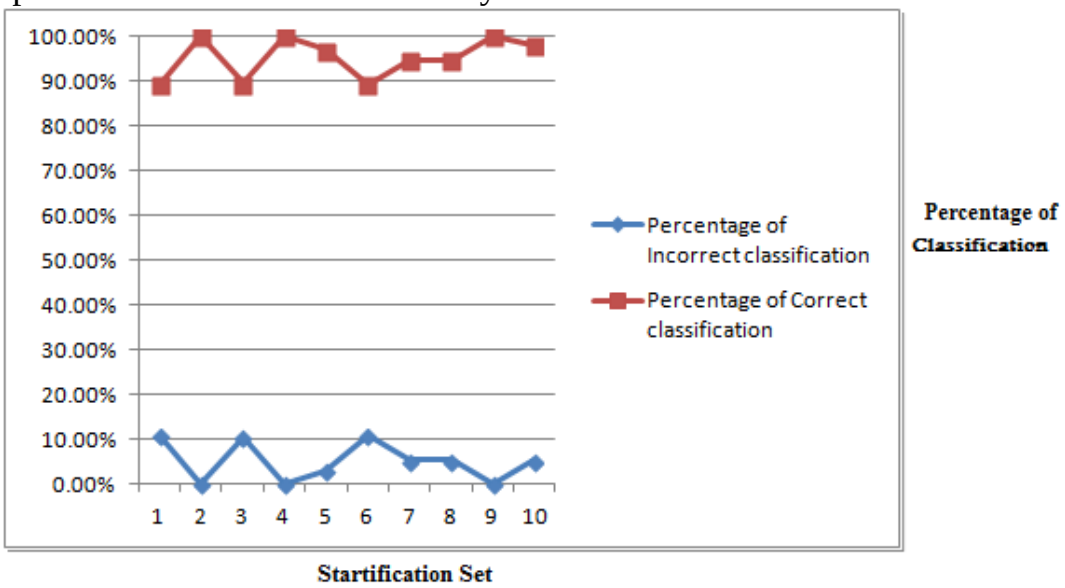


Figure 4.14. Result of mixed model on the subject of email

Table 4.22 Results of mixed model of the body of email

Stratification sets	1	2	3	4	5	6	7	8	9	10
Percentage of Correct classification	98.3%	93.3%	95%	100%	95%	95%	96.7%	98.3%	96.7%	93.3%
Percentage of Incorrect classification	1.7%	6.7%	5%	0%	5%	5%	3.3%	1.7%	3.3%	6.7%
Overall accuracy	96.2%									

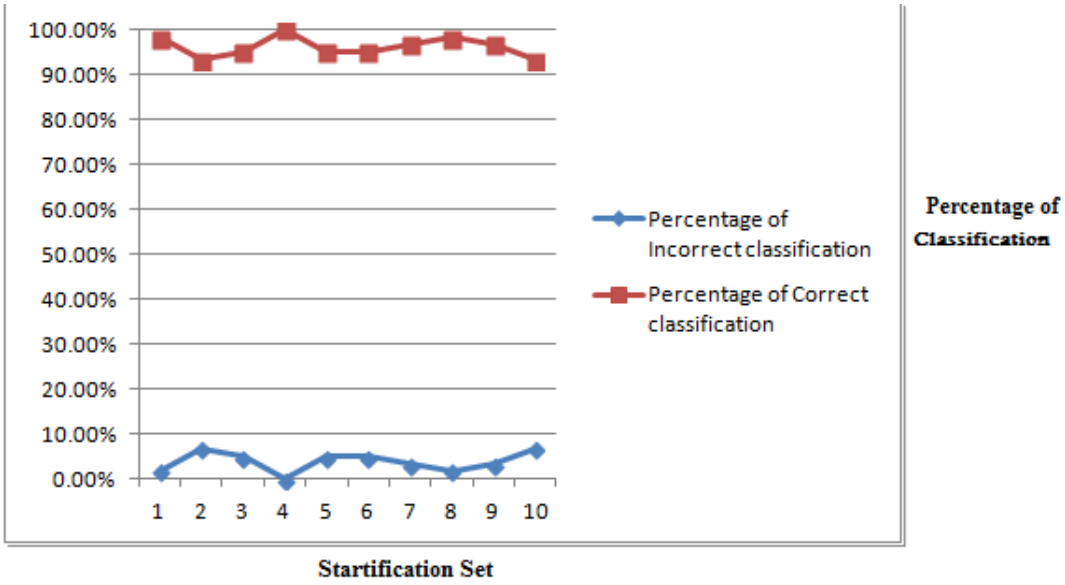


Figure 4.15. Results of mixed model of the body of email.

4.6.6 Discussion of Results for Mixed Spam Detection

The result of model showed that the Naïve Bayesian algorithm is the most effective way to use in detecting mixed (Arabic and English) emails. The mixed spam detection based on the collection of Arabic and English dataset. The model used different preprocess for each dataset and that effect on the end result. The results showed that the applied to the mixed detection model of the body of the message gave an accuracy of 96.2% of the English messages and Arabic messages.

The result of model showed that the Naïve Bayesian algorithm is the most effective way to use in detecting mixed (Arabic and English) emails. The model used different preprocessor foreach dataset. The results showed that the applied to the mixed detection model of the body of the message gave an accuracy of 96.2%.

4.7 Comparison between Models and various Classifiers

The CSV file is generated from the corpus and presented in the Excel file. The resultant classify is tested on Decision Tree J48, Logistic Regression and ZeroR classification techniques on 10-fold cross validation and tabulated in Table 4.23.

We experimented with three classifiers Decision Tree J48, Logistic Regression and ZeroR. The best results we obtained using proposed English model as shown in Table 4.23.

Table 4.23 Comparison between English model with various classifiers

Classifier	**Body Section**		**Subject Section**	
	Correctly Classified Instances in %	Incorrectly Classified Instances in %	Correctly Classified Instances in %	Incorrectly Classified Instances in %
English Model	96.2	3.8	91.5	8.5
J48	92.5	7.5	80.2	19.8
ZeroR	66.7	33.3	66.7	33.3
Logistic	79.7	20.3	82.9	17.1

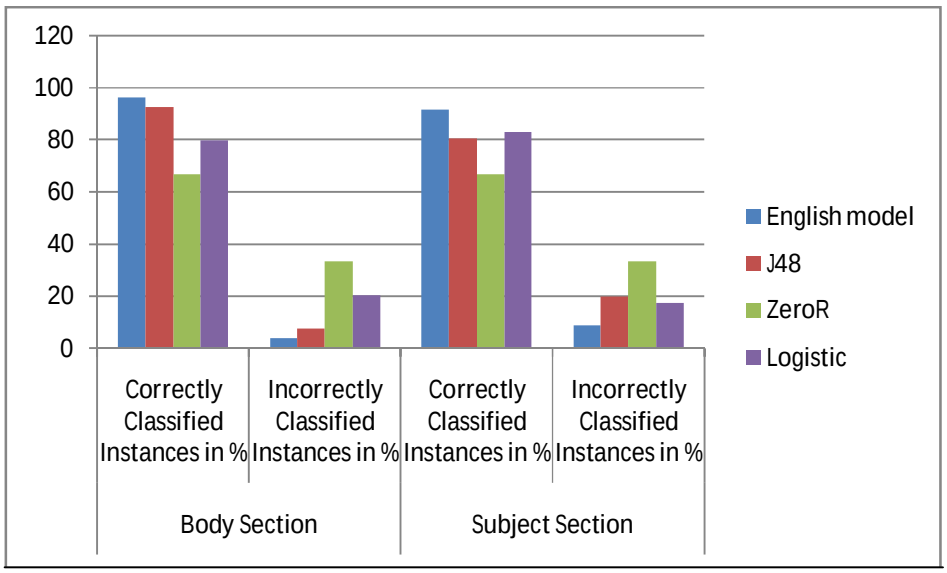


Figure 4.16 Performance of English model comparison

The results from Table 4.23 indicate English model classification gives better performance than the other three classifiers. Figure 4.16 shows the performance curves of English model, Decision Tree J48, ZeroR, and Logistic Regression classification techniques. The success criteria for text classification have significantly increased by using the proposed English detection model. The results showed an English spam detection model using Naïve Bayesian which yielded an accuracy of 96.2%.

Table 4.24 Comparison between Arabic model with various classifiers

Classifier	**Body Section**		**Subject Section**	
	Correctly Classified Instances in %	Incorrectly Classified Instances in %	Correctly Classified Instances in %	Incorrectly Classified Instances in %
Arabic Model	95.8	4.2	94.7	5.3
J48	92.8	7.2	92.8	7.2
ZeroR	92.8	7.2	92.8	7.2
Logistic	93.8	6.2	93.8	6.2

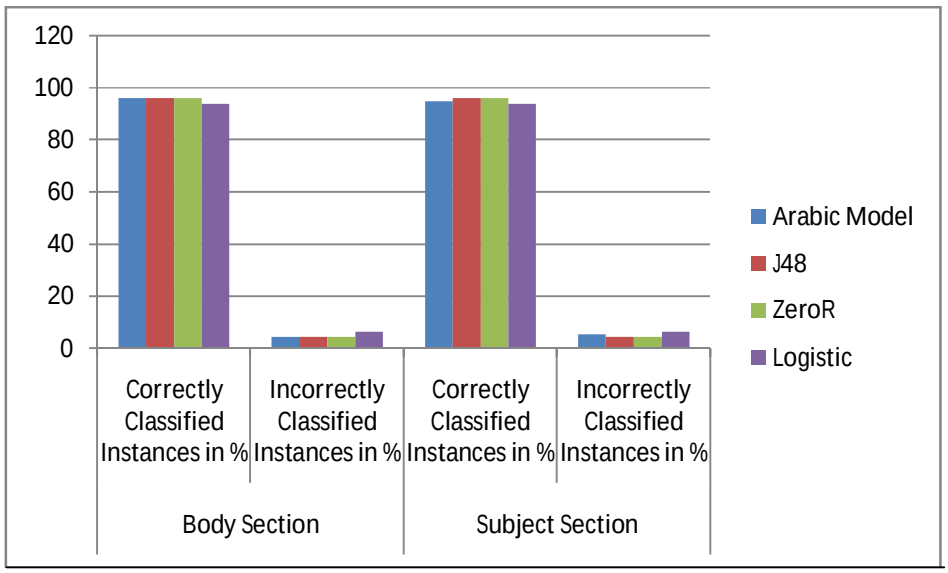


Figure 4.17 Performance of Arabic model comparison

The results from Table 4.24 indicate Arabic model classification gives better performance than the other three classifiers. Figure 4.17 shows the performance of Arabic model, Figure 4.17 shows the performance curves of Arabic model, Decision Tree J48, ZeroR, and Logistic Regression classification techniques. The success criteria for text classification have significantly increased by using the proposed Arabic detection model. The results showed an Arabic spam detection model using Naïve Bayesian which yielded an accuracy of 95.8%.

Table 4.25 Comparison between Mixed model with various classifiers

Classifier	**Body Section**		**Subject Section**	
	Correctly Classified Instances in %	Incorrectly Classified Instances in %	Correctly Classified Instances in %	Incorrectly Classified Instances in %
Mixed Model	96.2	3.8	90.2	9.8
J48	93.7	6.3	78.5	21.5
ZeroR	66.7	33.3	66.7	33.7
Logistic	83.2	16.8	84.7	15.3

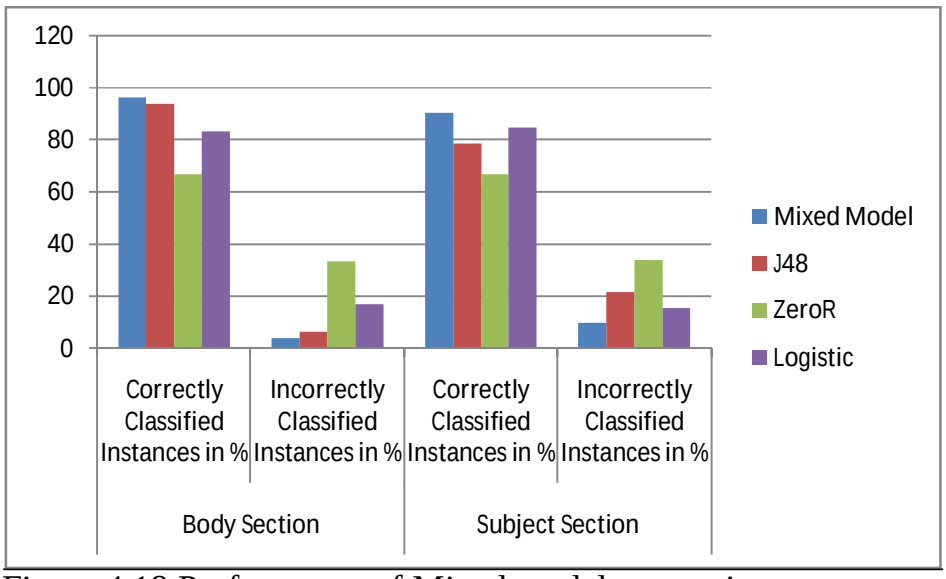


Figure 4.18 Performance of Mixed model comparison

The results from Table 4.25 indicate English and Arabic model classification gives better performance than the other three classifiers. Figure 4.18 shows the performance curves of English and Arabic model, J48, ZeroR, and Logistic Regression classification techniques. The success criteria for text classification have significantly increased by using the proposed English and Arabic detection model. The results showed an Arabic and English spam detection model using Naïve Bayesian which yielded an accuracy of 96.2%. The final results show that proposed model achieves high levels of accuracy. In addition, it can minimize the number of legitimate emails that are misclassified and is also able to detect a high number of spam messages. Nevertheless, several points of discussion are important regarding the suitability of the proposed method. It is also important to consider efficiency and processing time. Our system compares each email against a big dataset.

CHAPTER 5: PERSONALIZED SPAM DETECTION ALGOITHM

CHAPTER FIVE
PERSONALIZED SPAM DETECTION ALGORITHM

This chapter presents a personalized spam detection algorithm, which is based on the behavior of the user towards different types of message content [109].
The classification of the message is dynamically made based on what each user is like. The personalized email is abbreviated as Permail it is the web based spam detection system.
In this chapter we also present personalized spam classification model build on MATLAB [12].

5.1 Personalized Spam Detection System

Personalized spam detection algorithm integrates spam solution methods (whitelist, blacklist, keyword, and content filter) and web based solutions to classify the message.

5.1.1 System Architecture

The incoming email message passes through levels of classification and ends up in either the mail is spam or non-spam as shown in Figure 5.1.

A. Whitelisting

Initially, it has a set list of email addresses user "whitelist" ensures that all his important email addresses and web addresses will never be blocked. Most user whitelists consist of major legitimate newsletters and email address books. Messages that pass the filters be added to the whitelist, if they are not already there[110].

B. Blacklist

Initially, it has a set list of email addresses. Any message that failed to pass the whitelist level and then sent to this level will be checked against the blacklist, which contains email addresses that users never accepted email email addresses of the users who aren't accepts messages from [111]. If there is a match, the message will be sent to the junk mail.

C. Keyword filtering

By creating keyword lists, messages can be filtered based on a variety of words, phrases, and sentences extracted from the message header (subject).

[112]An example of a general type of keywords list is suggested by [113] and is presented in Appendix B.1. In our proposed listing a keyword list is created for each user and is populated and updated by keywords extracted from the messages identified as junk mail at the following level.

D. Content filtering

The proposed content filter acts on the content, the information contained in the mail body, to classify, accept or reject a message [33-35]. A user content filter is created for each user based on his behavior which requires information to be compiled and maintained for each user.

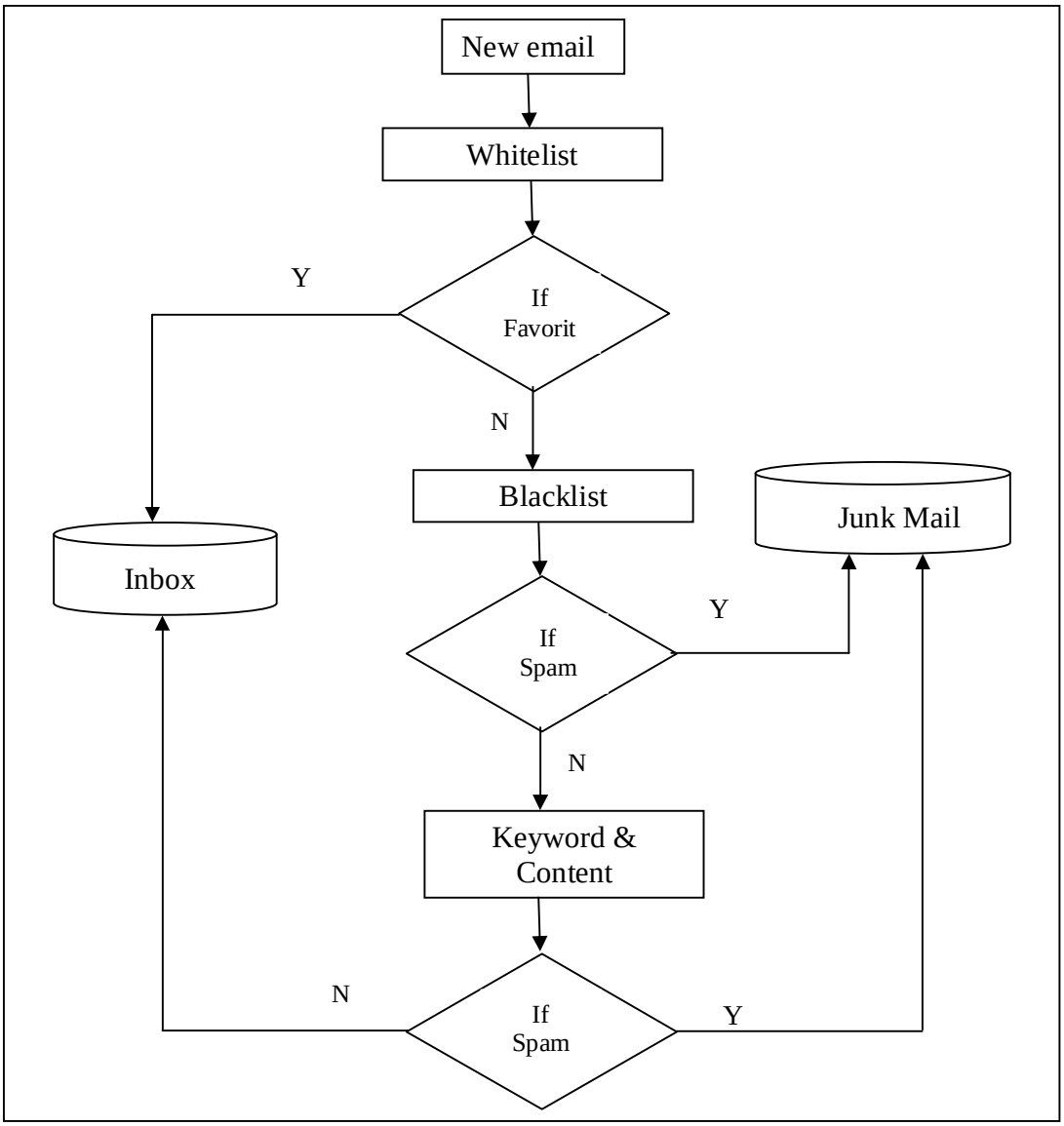


Figure 5.1 Flow work of personalized spam detection model

5.1.2 System Description

This section describes personalized spam detection algorithm and web page application as follow:

5.1.2.1 Personalized algorithm

The algorithm initially contains three lists:

- Whitelist: It contains Favorite email addresses (family, friends, related…. Etc.) Which are different from user to the other.
- Blacklist: At the beginning contains some spam email addresses and it will change dynamically with each user according to what he want to receive email from.
- Vocabulary list: It contains standard spam words each user can add or remove from the list to be as he like and the list will contain only the spam words.

The next paragraph presents the two main algorithms:

(1) When the user opens the Inbox

- If the user deletes the email

  The email address is added to the black list.

- If the user reads the email

  If the user then clicks delete

  Address candidate to be blacklist, after satisfying blacklist criteria, add an email address to the blacklist

- If the user then clicks, spam button then he is given the following choices:

  Title: processed spamming words added to Vocabulary list. Or,

  Words: user clicks the particular word. Or,

  Whole body: body processed spamming words sent to the vocabulary list.

Title process: selects spam words from the title and adds to Vocabulary list. Or,
Words process: selects the particular word from the title and adds to Vocabulary list. Or,
Whole body: selects all body and adds to Vocabulary list.

Now vocabulary list will acquire the words that this particular user classified as spam.

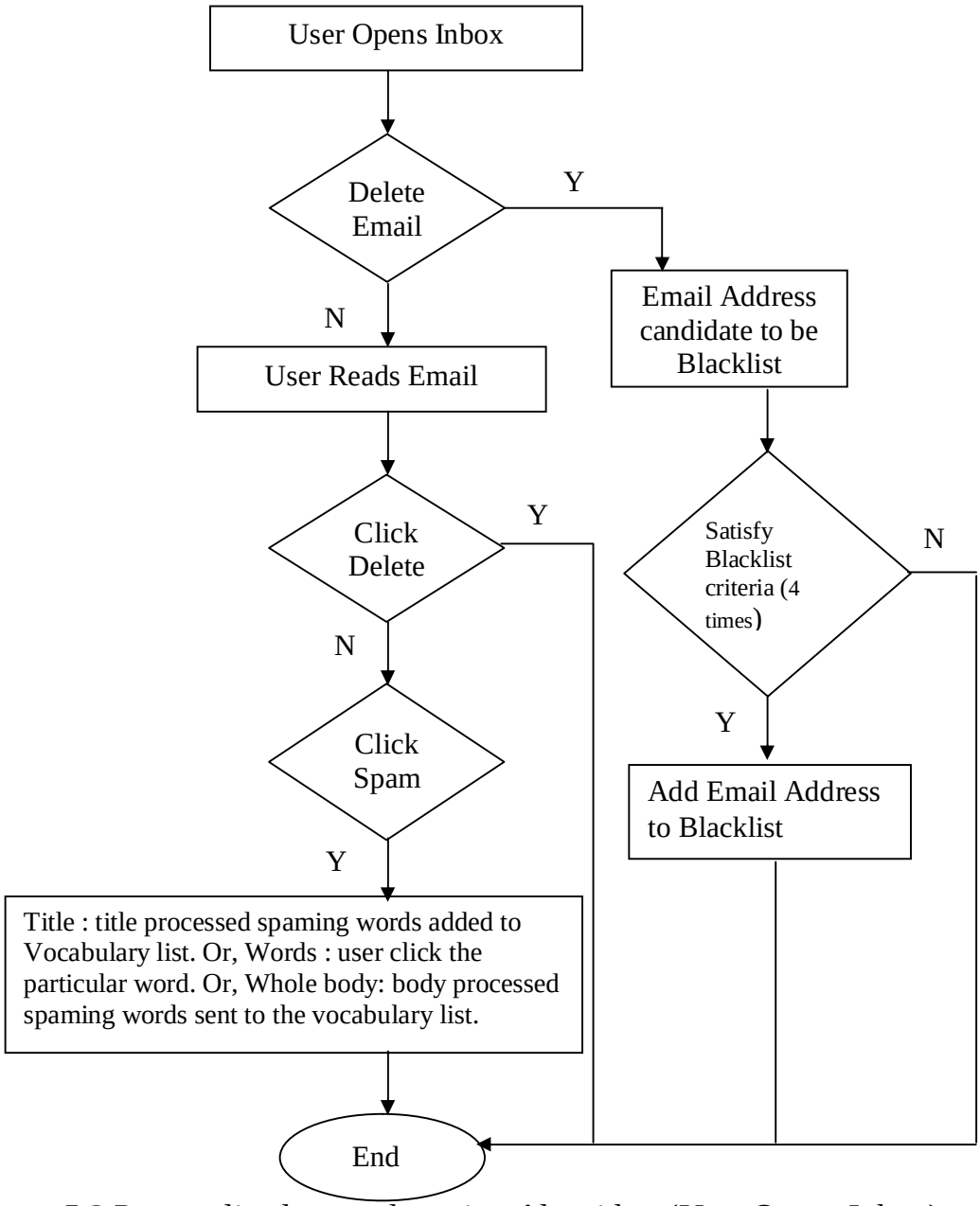


Figure 5.2 Personalized spam detection Algorithm (User Opens Inbox)

(2) The user reviews the junk list

- When the user selects a certain junk email and move it to the Inbox:

  Delete email address from the blacklist, Add to whitelist.
- When a user selects a certain junk email and clicks not spam:

  The email address is deleted from the blacklist.

- When a user chooses words (which the user consider legitimate and not spam). He clicks the words:

  Words are deleted from the vocabulary list.

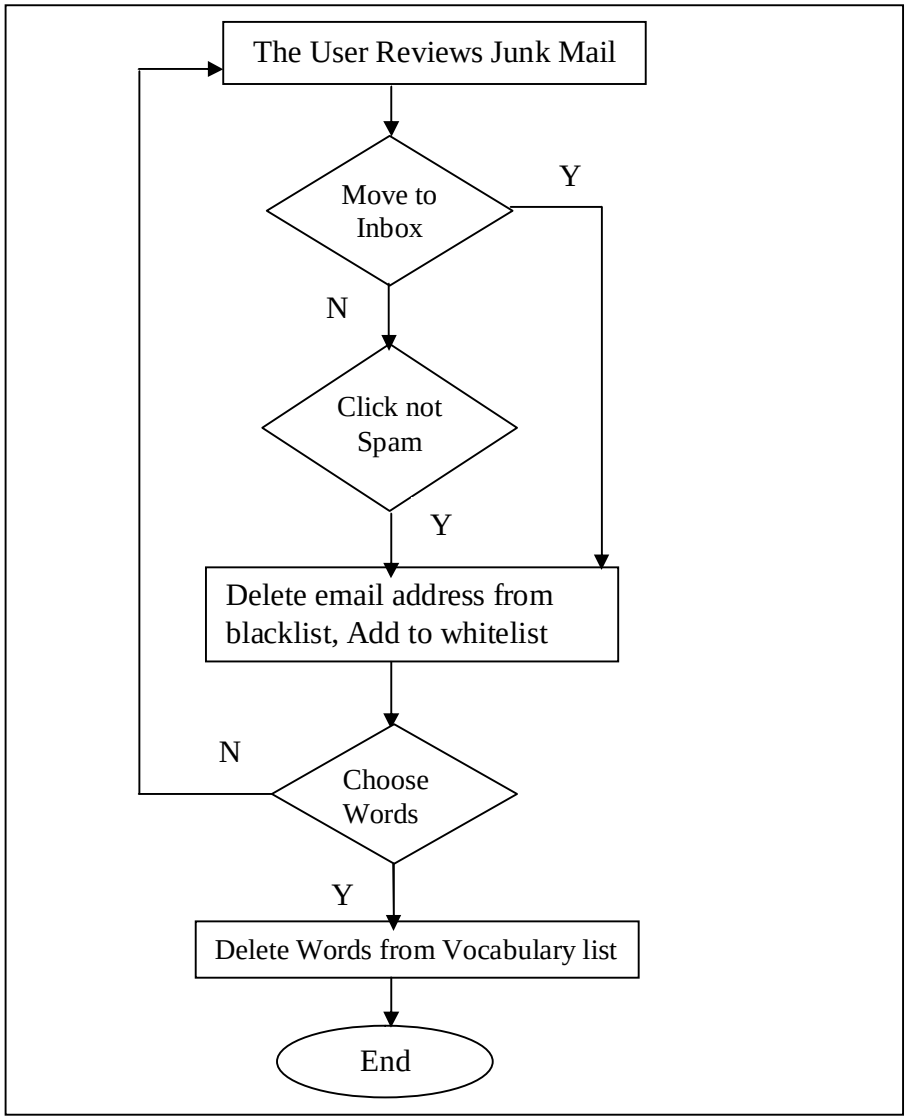


Figure 5.3 Personalized Spam Detection Algorithm (User Reviews Junk Mail)

The vocabulary list is dynamically changed to resemble the liking of each individual user. I.e. the vocabulary list is different from a user to the other. Hence, this is a personalized spam detection algorithm.

5.1.2.2 Personalized web pages

We attempted to build web based spam detection email system based on the user and his personalized like on websites and emails.

5.1.2.2.1 Permail database

We used a Microsoft Access database to build the system database consisting of about 15 tables the database schema is shown in Figure 5.4 and the database relationship as shown in Figure 5.5. The following is the list of database tables.

Candy Table		
PK	Cdid	Number
	bladd	Text

Userbl Table		
PK	uid	Number
Pk	blid	Number

Usermsg Table		
PK	Uid	Number
PK	Msid	Number

Junk Table		
PK	Jid	Number
	From	Text
	To	Text
	edate	Date/time
	subject	Text
	body	Memo

Message Table		
PK	mid	Number
	from	Text
	To	Text
	edate	Date/time
	subject	Text
	body	Memo

User Table		
PK	uid	Number
	fname	Text
	lname	Text
	bod	Date/time
	mobile	Text
	job	Memo
	country	Text

Userjn Table		
PK	uid	Number
PK	jid	Number

Bvocab Table		
PK	jid	Number
PK	vovid	Number

Wvocab Table		
PK	vocid	Number
PK	Msid	Number

Userwl Table		
PK	Uid	Number
PK	Wild	Number

Whitelist Table		
PK	Wild	Number
	wladd	Text

Blacklist Table		
PK	Blid	Number
	bladd	Text

Vocab Table		
PK	Vocid	Number
	Word	Text
	Type	Text

Login Table		
PK	Uid	Number
	Login	Text
	Pwd	Text

Candadd Table		
PK	Cdid	Number
	Isdel	y/n

Figure 5.4 Database tables of Permail system

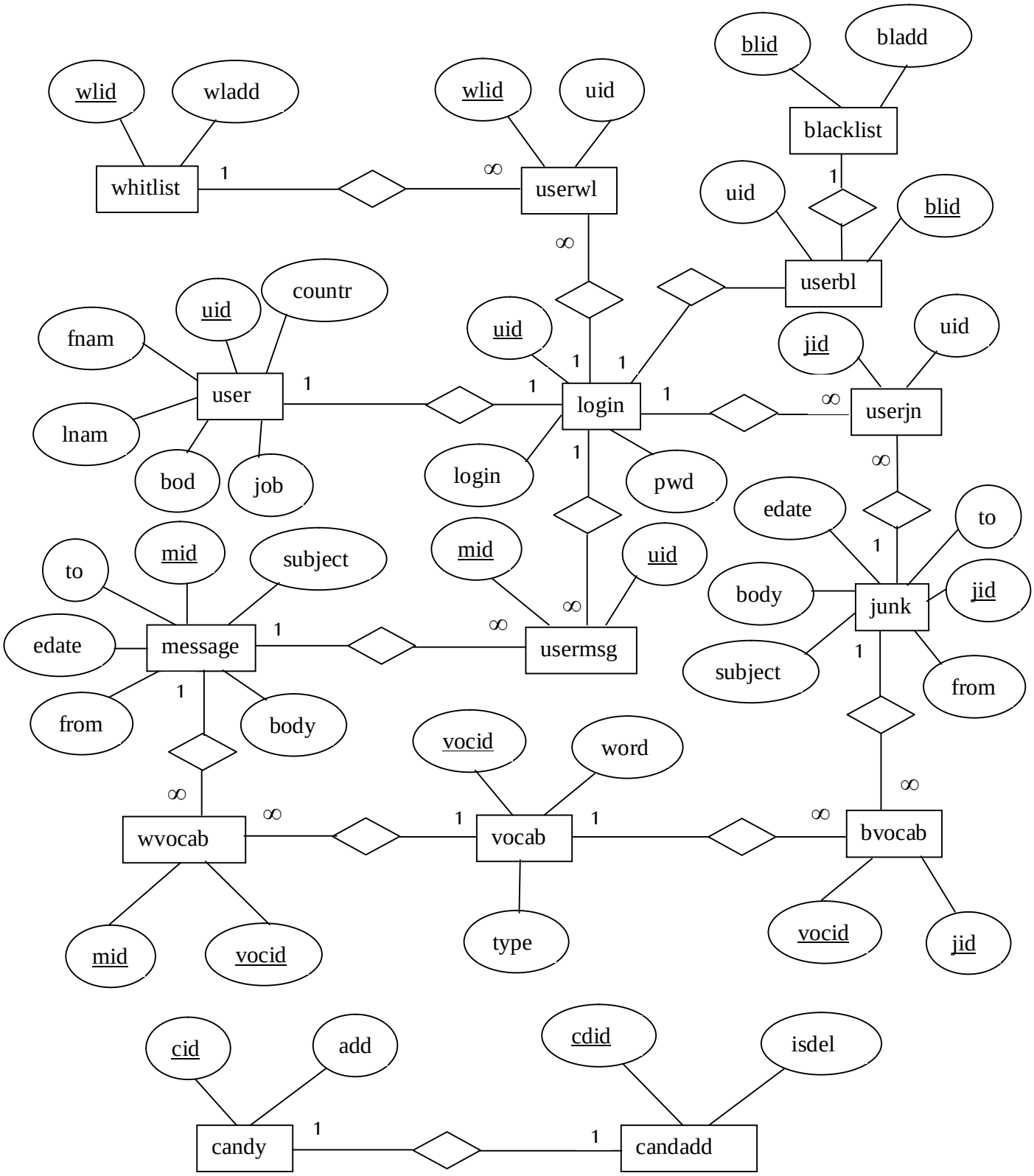


Figure 5.5 Database relationship of Permail system

5.1.2.2.2 Permail description

This section explains Permail web application that provides personalized features on the email build. Figure 5.6 shows the sitemap.
All sections discussed above have been developed and the source code of all programs on in the Appendix C.1.

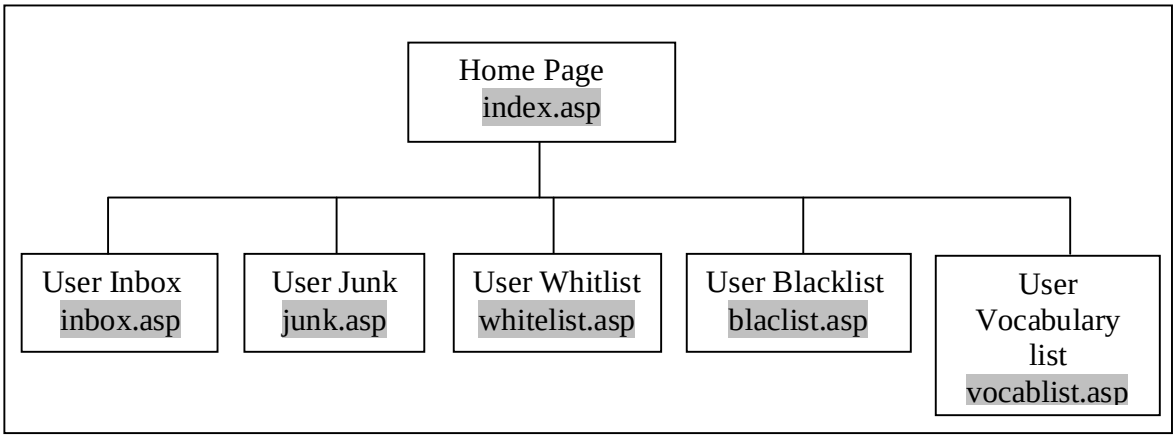


Figure 5.6 Permail sitemap

We developed web based application using ASP [14] this technology gives us ability to insert, update and retrieve data from database on the web page. We used ODBC [18] to create the connection between the database and the web pages.

All figures will mention next have been developed in the Appendix D. The home page is shown in Figure D.1 gives the ability to enter the mail system. When the user clicks on user login he will go into the login page as shown in Figure D.2 and if he is already registered he can enter his user name and password. Otherwise, he can create a new account as shown in Figure D.3 by entering his full information on the form. The following sections show the main processes of the web pages:

A. User Inbox

There are many users account created in the Permail. Each user has his different Inbox list message, whitelist, blacklist and vocabulary list and each list changes dynamically. When the user enters the correct login ID and password Permail open user Inbox page and this page contains many functions:

- If the user wants to view the message this page retrieves all messages from user Inbox. Figure D.4 shows different Inbox users a, b and c these users have a different Inbox.
- If the user clicks on the subject of any message he can read this message as shown in Figure D.5. After the user reads the message page, he can do the following:
    - Delete it by click on the delete button as shown in Figure D.6 or
    - If a user classifies this message as spam he can click on spam button and he will go to spam message as shown in Figure D.7 then he must select words to add into the vocabulary list:
        - Either from the title of the message as shown in Figure D.8 or
        - Select some words from body as shown in Figure D.9 or
        - Add all body to the vocabulary list as shown in Figure D.10.

B. User junk

User junk mail page retrieves all junk mail and contains many functions as follows:

- If the user wants to view the message he can retrieves all messages from the user's junk mail as show in Figure D.11 shows different users.
- If the user knows some message is not junk mail he can clicks to move the message to Inbox as shown on Figure D.12 and email address will be deleted from blacklist and added to the whitelist.
- If the user clicks on the subject of any message he can read this message as shown in Figure D.13. After the user reads the message page, he can do the following:
    - Delete it by click on the delete button or

- If the user finds this message not spam he can click on not spam button and go to the page shown in Figure D.14 to select the non-spam words from:
    - If the words of the title as shown in Figure D.15, the user can select these words and delete from blacklist and add to the whitelist or
    - If the words on the body as shown in Figure D.16, the user can select these words and delete these words from blacklist words and add to the whitelist or
    - If all the words of the body are non-spam words as shown on Figure D.17, the user can select all the body and delete these words from blacklist words and add them to the whitelist.

C. User list

This section presents user whitelist, blacklist and vocabulary lists. Figure D.18 (a, b and c) shown the different whitelist between users a, b, and c.

Figure D.19 (a, b and c) shown the different blacklist between users a, b, and c.
Figure D.20 (a, b and c) shown the different vocabulary between users a, b, and c.

5.1.3 Advantages of Personalized Spam Detection

Table 5.1 Advantages and disadvantages of mail services

Mail service	Advantages	Disadvantages
Permail	- Personalize performance. - Body spam list. - User dynamic whitelist email address. - User dynamic blacklist email address. - User dynamic white vocabulary list.	- Used locally.

	- User dynamic black vocabulary list.	
Hotmail		- All users share same spam blacklists.
Gmail	- Bayesian	- All users share same spam blacklists.
Yahoo Mail	- Spamgraud feature. - No false positive.	- All users share same spam blacklists.

5.1.4 Permail experiment

This section compared Yahoo Mail, Hotmail, and Gmail with proposed Permail to test to his features and advantages.

This section compared Yahoo Mail, Hotmail, and Gmail with proposed Permail.

- We created three email accounts on Yahoo Mail, Hotmail, and Gmail.
- We created suspicious email accounts on the top free email such as outlook.sa, mail.com, gmx.com, shortmail.com, myway.com and aim.com.
    - abc.dd@aim.com
    - Freeone98@gmx.com
    - buyone@shortmail.com
    - Topten10@myway.com
    - Abc.abc2@outlook.sa
    - Shop123@mail.com
    - Free.mony@lycos.com
- We used above emails to send a message to Yahoo Mail, Hotmail, and Gmail, we used email corpus gets from the Second Conference on Email and Anti-Spam CEAS 2005, Stanford University, Palo Alto, CA [15].

- We used valid mail accounts on Gmail, Yahoo Mail and Hotmail (my own accounts) to send spam messages.
- We sent equal numbers of spam and non-spam emails for all mail services.
- All three valid mails allow send spam message, that means there is no filter to send and they only filter the receive spam emails.

Comparison aimed to count the amount and percentage of spam and non-spam that showed up in the accounts' inboxes.

Table 5.2 Mail service with spam in the Inbox

Mail service	Total Inbox	Non-spam Inbox	Spam % of Inbox	Non-spam % of Inbox
Permail	32	30	6.25%	93.75%
Gmail	54	24	55.66%	44.44%
Hotmail	59	23	61.02%	38.98%
Yahoo	72	32	55.66%	44.44%

Table 5.3 Mail service with spam in junk

Mail service	Total junk	Non-Spam in junk	Spam % of junk
Permail	72	2	2.77%
Gmail	48	40	83.33%
Hotmail	44	34	77.27%
Yahoo	30	0	0

Table 5.4 Mail service with false positive and false negative

Mail service	False negative	False positive
Permail	2	2
Gmail	30	8
Hotmail	36	10
Yahoo	40	0

Compare the classification results of Permail, Hotmail, Yahoo and Gmail it identify that the Permail is the best classifier to detect spam emails. The results indicated that the Permail web spam detection showed the less false positive.

5.2 Spam Classification Model Based on Naïve Bayesian

Since different users receive different types of legitimate emails, the process of the training of the algorithm would be different for different users. Certain words may have more occurrences in the legitimate emails of one user; however, for another user, these words may have more occurrences in spam emails. Therefore, in the training process of algorithm probabilities of spaminess of words are computed differently and as a result, we would have different classification for each user. This is needed the personalization of the model. Our proposed model is based on Naïve Bayesian (NB) algorithm[114].

5.2.1 Naïve Bayesian

As could be assumed by its name, the main idea of Naïve Bayesian spam classifier is the use of Bayes theorem which is used widely in the context of probability [115]. As such, assuming that given a specific word, we would like to find out what is the probability of being spam for a message which contains this word. From Bayes rule, we may write:

$$P(spam|word) = \frac{P(word|spam) \times P(spam)}{P(word|spam) \times P(spam) + P(word|ham) \times P(ham)} \quad .(1)$$

Where

P(spam|word) is the probability of being spam for a message which contains the specific word. Moreover,

P(spam) is the probability that a message is spam in general and is equal to the number of spam emails we have in our data set divided by the total number of emails.

P(word|spam) is the probability of occurrence of a specific word in spam messages. We find this value from our training set or we may check the frequency of occurring of this word in the spam emails.

P(word|non-spam) is the probability of occurrence of a specific word in non-spam message. Similarly, we find this value from our training set or we may check the frequency of occurring of this word in the non-spam emails. Finally,

P(non-spam) is the probability that a message is non-spam in general. This value is equal to the number of non-spam emails we have in our training set [115].

In the context of machine learning, the problem of classifying emails as spam or non-spam is translated into a classy set of objects (emails in here).

For each object (email), we have a feature vector. For the emails, the components of feature vectors are words. We choose these words according to our training set. We find out the most frequently occurring words in our training set. Then we form a vector size.

For each new email that we would like to classify as either spam or non-spam email we first form its corresponding feature vector such that we put 1 for each word which is contained in the message and 0 for each word which is not contained in the message.

A useful preprocess is to consider the message body and try to consider features which help us to make the algorithm more efficient, we can consider the words which belong to the same family as one word. For this aim, we can first convert the message to a new message in which words of a same family appear as one unique word. As an example, instead of words like “Dealer”, “dealing” and “deal”, we may only use one word like “deal” which is the root of this family [115].
All sections discussed above have been developed and the source code of all programs on in the Appendix C.2.

5.2.2 Training Phase

If we compare the spam emails and non-spam emails, we will find out that they are certain words, e.g. business related words, which have more occurrence in spam emails than in non-spam emails. The proposed model is to first provide a set of training examples containing both spam and non-spam emails. This set has been downloaded from the Second Conference on Email and Anti-Spam CEAS 2005, Stanford University, Palo Alto, CA [15]. Then the model will find out for each word the probability of occurring in spam and non-spam emails in that training set.

5.2.3 Testing Phase

As a result, when we have a new email which we would like to classify as either spam or non-spam, first we find out the most interesting components of that email (rather than pretty common words like: and, or, the, etc.). Afterwards, we find out the probability of those components or words with the values we found in the training process [116].

5.2.4 Rare Words

These words are the ones which cause the term 0/0 in the product term we had previously for computation of the probability of spaminess. Since these words are not available in our training set, there is no information about their occurrence in spam and non-spam emails. Therefore, we may discard these rare words when seeing them for the first time. In addition, we have neutral words like “a”, “an”, “the”, “some” and etc. which are very common in either spam or non-spam emails [117].

5.2.5 Functions of Algorithm

We used MATLAB to build proposed spam classification model. We list function with short description as is shown follow:

Vocalist = getVocabList(n)

This function loads the pre-defined vocabulary list. n indicates the length of our vocabulary list. This length-n vector is our general feature vectors where features in spam classifications are different words. We got this vocabulary list from Stanford online machine learning course computer assignments [118].

Posting = porterStemmer (inString)

This function converts the words which have the same stem and belong to the same family (e.g. dealing, dealer, deal) in one word which is the stem of the family. The algorithm is called Porter Stemming algorithm[119].

[vocabList vocabHist] processEmail(email_contents,vocabList,vocabHist)

This function converts each email into a vector of features (defined by "vocalist" function mentioned previously) and removes unnecessary characters such as punctuation marks. In addition, it returns the distribution of the words from "vocalist" function in the email which we would like to classify it. This function can update the vocabulary list if the algorithm faces a new word which is not available in our vocabulary list.

[Inmail]= indicateEmail(email_contents,vocabList,vocabScore)

This function indicates whether the given email is spam or a legitimate email based on the vocabulary list and the spamicity of the corresponding words of the message.

5.2.6 Spam Classification Model Interface

For better understanding of the process, we have designed GUI as shown on Figure 5.7. Through the GUI, users can provide the program with their own email and check whether they are given email is classified as either spam or legitimate email.
Figure 5.7 shows the screen that user can enter his message and click on classify to classify messages as spam or non-spam.

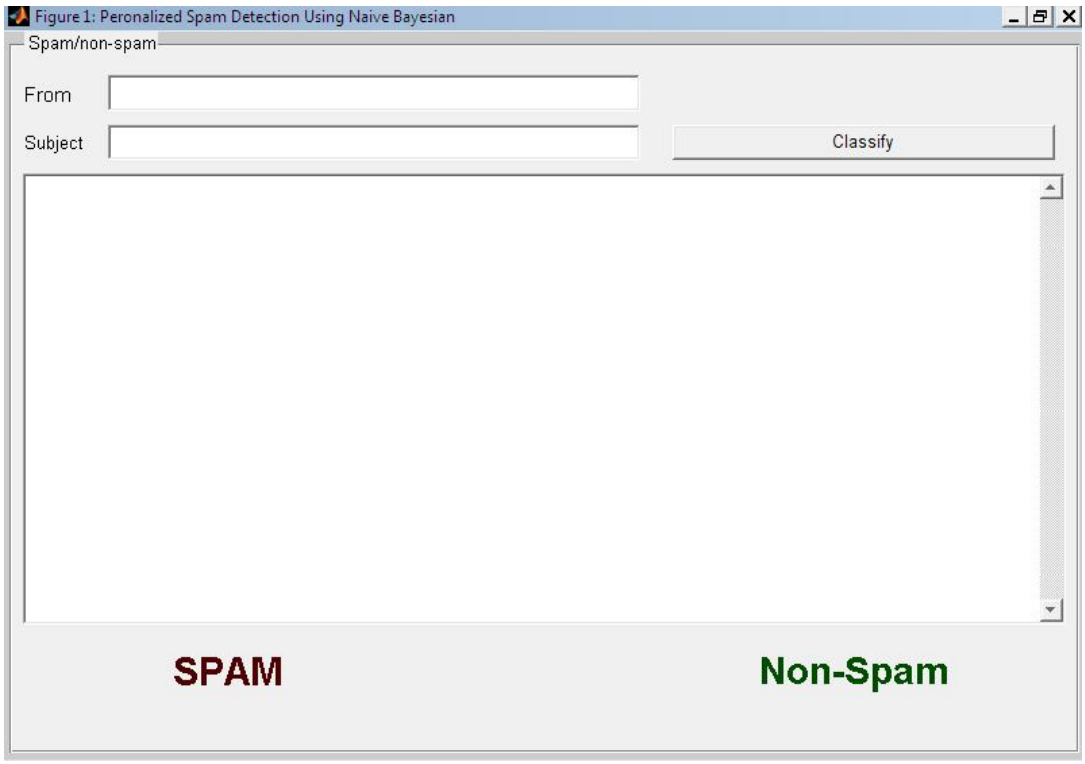


Figure 5.7 General interface of spam classification model

If a user writes message " dear, we give you big free wine gift" and write the sender address" freeabc@free.com" and subject "free gift", then click on classify button he will get the result shows that this message is spam as shown on Figure 5.8 change on the color of the spam word from the brown color to red color indicate the classification of the message.

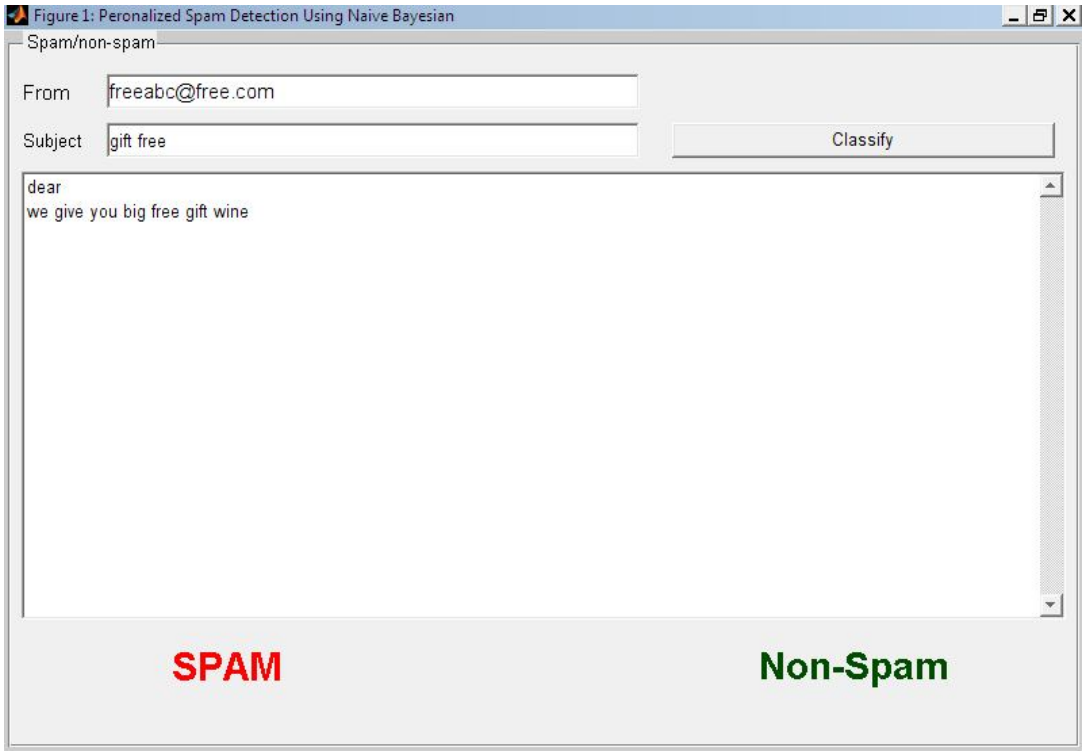


Figure 5.8 Interface to classify the spam email

If a user writes message " Eid Mubark mum" and write the sender address" shreef@yahoo.com" and subject "Eid Mubark", then click on classify button he will get the result shows that this message is non-spam as shown in Figure 5.9 when the color of the word non-spam will be white green color.

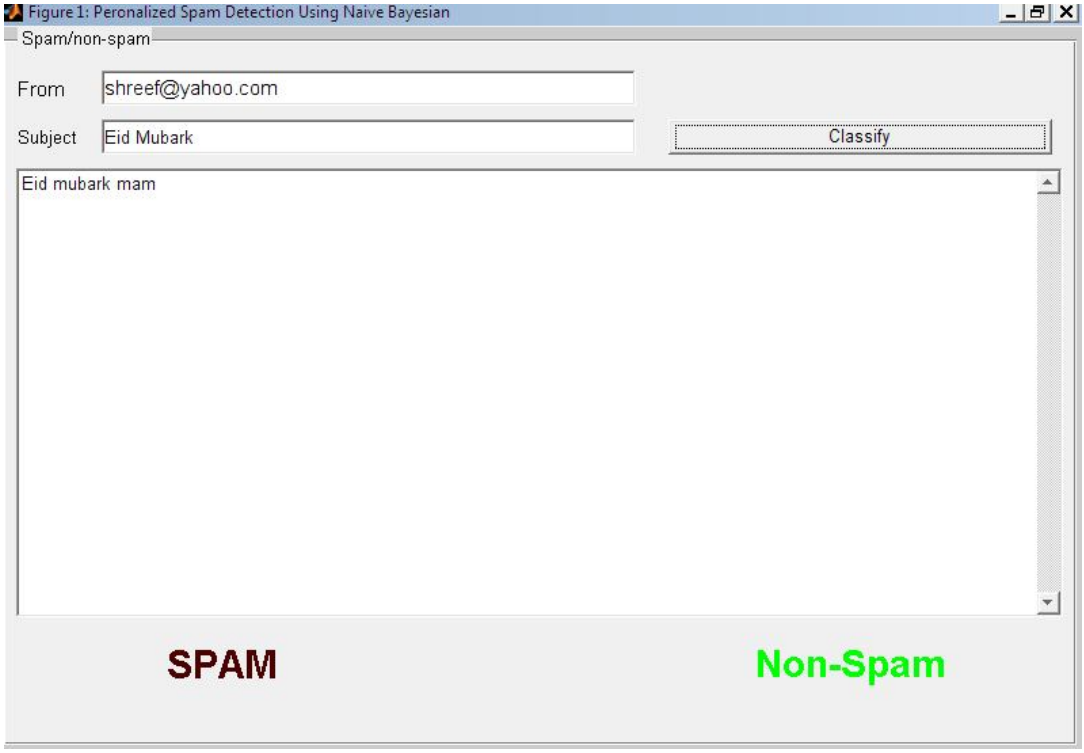


Figure 5.9 Interface to classify the non-spam email

5.2.7 Evaluation Matrix

For tasks like email classifying that amounts of spam and non-spam emails are different to each other, we have skewed classes. The metrics which we evaluate classification algorithms with skewed classes are as the following:

The metrics which evaluate classification algorithms are as the following:

True positive

Emails which are spam and we have classified them correctly.

True negative

Emails which are not spam and we have classified them correctly.

False positive

Emails which are not spam and we have classified them as spam incorrectly.

False negative:

Emails which are spam and we have classified them as non-spam incorrectly.

Precision:

Of all emails we have classified as spam, what fraction of them is really spam spamming emails?[120]

Here are the formulas of above metric:

$$Precision = \frac{true\ positive}{true\ positive\ +\ false\ positive} \ldots\ldots (2)$$

Recall:
Of all emails which are really spam, what fraction of them have been classified correctly by the algorithm? [121]

Here are the formulas of above metric:

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \quad \ldots\ldots (3)$$

We call the metric through which we can compare performance of different

Algorithms as F1 score. F1 scores are given by the following formula:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \ldots\ldots\ldots (4)$$

5.2.8 Evaluation Of Recall And Position

We have a large data set which we should use it for both trainings the algorithm and testing the algorithm.

Based on the fraction of the data set to which we dedicate to training set, we might see different performance of our algorithm.

The following Figure 5.10 shows the precision of the algorithm based on the percentage of data dedicated to the training set.

Table 5.5 Result accuracy of the classification model

Percentage of training set	Accuracy (%)
0.1	94.9
0.2	97.6
0.3	97.8
0.4	98.3
0.5	98.7
0.6	98.9
0.7	98.5
0.8	98.8
0.9	99

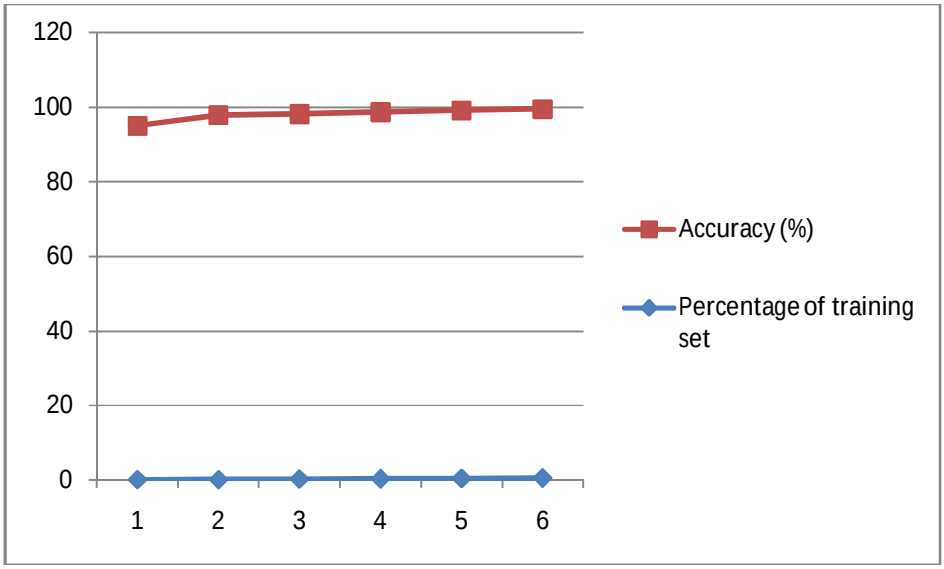


Figure 5.10 Precision of the classification model

The next Figure 5.11 is corresponding to the second recall metric.

Table 5.6 Result of Recall metric algorithm for classification model

Percentage of training set	Spam recall (%)
0.1	77.5
0.2	91
0.3	92
0.4	95
0.5	96.5
0.6	100

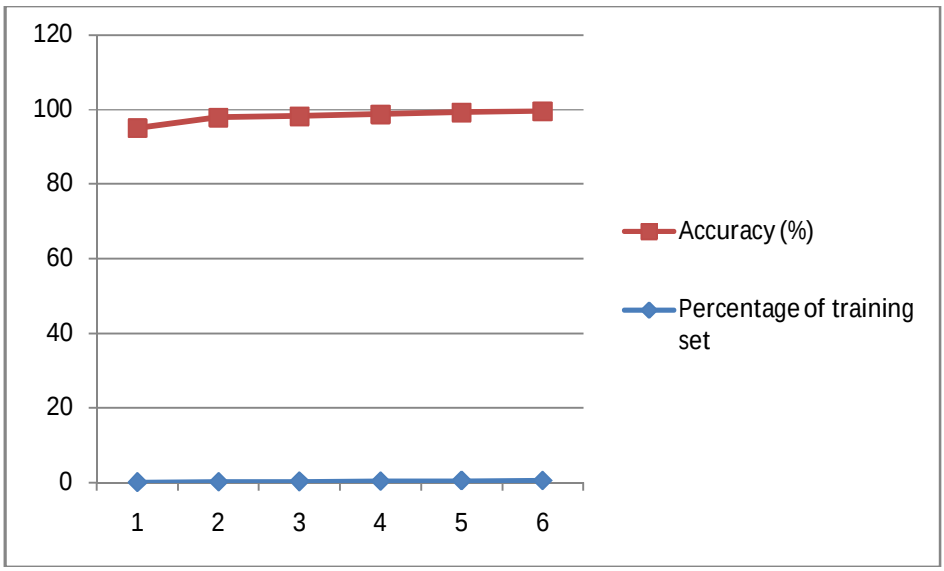


Figure 5.11 Recall metric algorithm for classification model

CHAPTER 6: CONCLUSION

CHAPTER SIX
CONCLUSION

This chapter consists of a summary of the thesis followed by a conclusion and recommendations for further research.

6.1 Summary

- The study was set out to build a user personalized spam detection model.
- The first model to detect English spam, second to detect Arabic and mixed (Arabic and English) spam emails
- And third personalized spam web based detection system, the reasons and motivation for spam detection models, the limitation of the resources for Arabic spam detection, no publish studies in this field, there is some problem on performance on the most of famous free mail systems and the negative economic effect of the spam problem.
- The study has also sought to know whether a proposed spam detection solution can result in effective compared with any mail system like Yahoo, Hotmail and Gmail, an algorithm used in the proposed Arabic spam detection can give better result than others algorithms.
- The study sought to collect and build an Arabic corpus for testing the Arabic spam detection models.

The main findings are summarized within the respective chapters: Arabic spam detection model and personalized spam detection algorithm. This section will synthesize the findings to obtain the study's four research objective.

1. Build a dynamic and personalized model to detect English spam emails and then test the model against a standard data set.
    - Design GUI to check the email classification.
    - Apply Naïve Bayesian (NB) to the proposed model.
    - Perform the preprocess.
    - Probabilities of spaminess of words are computed differently and as a result we would have different classification for each user.

- On the test phase, we find out the most interesting words of a message.
- Evaluated the model by using recall and precision Matrix algorithms [16].

2. Modified the English model to detect Arabic and mixed (English and Arabic) spam emails.
    - Extracted a set of features from the Arabic dataset. And performed feature selection.
    - Evaluated Naïve Bayesian algorithms to classify incoming email as spam or non-spam.
    - Used 10-fold cross validation to calculate the accuracy of the classifiers.

3. Collected and built an Arabic corpus for testing the Arabic spam detection model.
    - Built a larger dataset of Arabic emails include spam and non-spam. The first part is called a training dataset used to build the model, and contains around 1066 emails, 512 spam messages. While the second part is called a test data set, and contains around 554 non-spam messages, and used to evaluate the model.
    - Build an Arabic spam words which are used in spam filtering, it contains about 200 of foumase spam words.

4. Developed personalized spam detection web based (Permail) and compare the spam filtering capabilities of Microsoft Hotmail, Google Gmail, Yahoo Mail and Permail to determine the effectiveness of spam filtering for each provider. The key measurements for this mail system are the quantity and percentage of spam in the Inbox.
    - Build an email database.
    - Create a connection string.
    - Develop ASP program files that can perform the transactions between the front end (web pages) and back end (database).
    - Examined performance of personalized spam detection.
    - Compare the classification results of Permail, Hotmail, Yahoo and Gmail to identify the best classifier to detect spam emails.

One main point with existing theoretical on the spam detection and filtering was the truth of that most the published literature on the English email and there are no published detection models on Arabic messages. Currently this situation creates an unjustified disadvantage in the community of Arabic users. This thesis seems to point to the fact that Arabic spam detection has still not covered from the researchers and also in this thesis an attempt is made to extend the spam detection to dynamically follow the liking of the user. It is termed personalized spam detection [7]. This study has used data mining algorithm to show that the current published research and studies of spam detection and filter are not working on the Arabic emails. The theoretical arguments for this justification suggest the need for more algorithms and models which will enable researchers to work on this poor area.

6.2 Conclusion

This thesis has achieved the following:

- Provided a personalized spam detection mode and tested it.
- And compared it to the known filters : Yahoo,Hotmail and Gmail. The proposed model outperformed these known filters
- Built an Arabic spam corpus.
- Provided a personalized spam detection mode for Arabic messages and for mixed messages. Again, the proposed model outperformed the above mentioned known filters.

6.3 Recommendations for Further Work

- Evaluate the algorithm on live mail systems.
- Collect a larger dataset of Arabic emails (spam and legitimate) to use in the training Arabic model and build an international Arabic email corpus and Arabic spam words which can be used in Arabic spam filters.
- Use multi-level classification based on another technique (such as ANN, SVM, etc.) to build the spam detection model.
- Build international Arabic spam detection system.

REFERENCES

[1] J. B. Postel, “Simple Mail Transfer Protocol,” 1982.

[2] J. Myers, “Post Office Protocol,” 1996.

[3] M. R. Crispin, “Internet Message Access Protocol,” 2003.

[4] P. Sawers, “The Origin Of The Word Spam,” 2010.

[5] M. B. Prince, W. Adams, L. Holloway, E. Langheinrich, B. M. Dahl, and A. M. Keller, “Understanding How Spammers Steal Your E-Mail Address : An Analysis of the First Six Months of Data from Project Honey Pot,” no. March, 2003.

[6] S. Hershkop and S. J. Stolfo, “Identifying spam without peeking at the contents,” Crossroads, vol. 11, no. 2, pp. 3–3, Dec. 2004.

[7] A. Ibrahim and I. M. Osman, “A Behavioral Spam Detection System,” pp. 77–81, 2011.

[8] “World Summit on the Information Society [Online] Availabe :http://www.itu.int/wsis/implementation/igf/ , [Accessecd,” no. March, p. 2013, 2013.

[9] B. Stone, “Spam Doubles , Finding New Ways to Deliver Itself,” p. 404482, 2006.

[10] L. Y. A. Ve, P. A. L. O. A. Lto, and C. A. T. E. L. F. Ax, “Email Statistics Report ,” pp. 2009–2013, 2013.

[11] H. A. Wahsheh and M. N. Al-kabi, “Analyzing the Popular Words to Evaluate Spam in Arabic Web Pages,” vol. II, no. Ii, pp. 22–26.

[12] E. W. Kamen and B. S. Heck, “Fundamentals of Signals and Systems Using the Web and MATLAB, ” ISBN-10: 0131687379, 3rd Edition, Pearson Edtion Intrnational,2006 .

[13] B. S. Bird, E. Klein, E. Loper, and C. S. Bird, “Natural Language Processing with Python,” p. 95472, 2009.

[14] U. M. Frontpage and C. Asp, Using Microsoft FrontPage to Create ASP pages For example ... 2000.

[15] G. Cormack and T. Lynam, “Spam Corpus Creation for TREC,” pp. 0–1, 2005.

[16] N. Nakashole, M. Theobald, and G. Weikum, “Scalable Knowledge Harvesting with High Precision and High Recall,” no. 1955, 2010.

[17] S. Kale, R. Kumar, and S. Vassilvitskii, “Cross-Validation and Mean-Square Stability.”

[18] B. Ripley, “ODBC Connectivity,” Source, pp. 1–34, 2012.

[19] J. Hovold, “Naive Bayes Spam Filtering Using Word-Position-Based Attributes.”

[20] L. De Ferrari and S. Aitken, “Mining housekeeping genes with a Naive Bayes classifier.,” BMC Genomics, vol. 7, p. 277, Jan. 2006.

[21] S. Dumais, “A Bayesian Approach to Filtering Junk E-Mail,” no. Cohen, 1996.

[22] J. S. Park and H. Lu, “Anti-Spam Approaches : Analyses and Comparisons Anti-Spam Approaches : Analyses and Comparisons,” pp. 36–47, 2009.

[23] R. Kamboj, “A Rule Based Approach for Spam Detection,” 2010.

[24] C. Lucas, “An Overview of Spam Blocking Techniques.”

[25] S. Hird, “Technical Solutions for Controlling Spam,” pp. 4–6.

[26] “Blacklist vs . Whitelist.” http://www.processor.com/articles/P2724/33p24/33p24chart1.pdf?guid

[27] I. Santos, C. Laorden, X. Ugarte-pedrero, B. Sanz, and P. G. Bringas, “Spam Filtering through Anomaly Detection.”

[28] A. Ko, A. Chowdhury, and J. Alspector, “The Impact of Feature Selection on Signature-Driven Spam Detection.”

[29] D. S. Eth-tutor, B. T. Supervisor, and B. Plattner, “Signature-based Extrusion Detection,” no. August, 2008.

[30] J. Yan and P. Cho, “Enhancing Collaborative Spam Detection with Bloom Filters,” 2006 22nd Annu. Comput. Secur. Appl. Conf., pp. 414–428, Dec. 2006.

[31] V. P. Deshpande, R. F. Erbacher, and C. Harris, “An Evaluation of Naïve Bayesian Anti-Spam Filtering Techniques,” 2007 IEEE SMC Inf. Assur. Secur. Work., no. June, 2007.

[32] B. V. P. Deshpande, “An Evaluation Of Filtering Techniques In A Naïve Bayesian Anti- Spam Filter,” p. 2004, 2004.

[33] A. Schwartz, “SpamAssassin,” 2008.

[34] J. Carpinter and R. Hunt, “Tightening the net : a review of current and next generation spam filtering tools,” 1978.

[35] J. Basilico and T. Hofmann, “Unifying collaborative and content-based filtering,” Twenty-first Int. Conf. Mach. Learn. - ICML ’04, p. 9, 2004.

[36] P. Graham, “Stopping Spam,” no. August, 2003.

[37] P. Taninpong and S. Ngamsuriyaroj, “Incremental Adaptive Spam Mail Filtering Using Naive Bayesian Classification,” 2009 10th ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput., pp. 243–248, 2009.

[38] Y. Chen, C. Lu, C. Huang, and P. R. A. Pproaches, “Anti-Spam Filter Based on Naïve Bayes ,” pp. 1–5, 2009.

[39] H. Katirai, W. Ontario, L. Programming, and A. I. Group, “Filtering Junk E-Mail :,” 1999.

[40] K. R. Gee, “Using latent semantic indexing to filter spam,” Proc. 2003 ACM Symp. Appl. Comput. - SAC ’03, p. 460, 2003.

[41] C. O. Brien and C. Vogel, “Spam Filters : Bayes vs . Chi-squared ; Letters vs . Words,” no. September, 2002.

[42] V. Metsis, “Spam Filtering with Naive Bayes – Which Naive Bayes ?,” 2006.

[43] T. Stone, “Parameterization of Na ¨ ıve Bayes for Spam Filtering,” 2003.

[44] A. Khalid, “A Multi-Phase Feature Selection Approach for the Detection of SPAM,” vol. 1, no. 3, pp. 96–99, 2011.

[45] S. Braganza, “Variable Thresholding In Naïve Bayesian Spam Filters.”

[46] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos, “An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages,” Proc. 23rd Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR ’00, pp. 160–167, 2000.

[47] M. Gridach and N. Chenfour, “Developing a New Approach for Arabic Morphological Analysis and Generation 2 . Overview of Arabic morphology,” no. 1.

[48] M. Park, “CROSS-DIALECTAL ACOUSTIC DATA SHARING FOR ARABIC SPEECH RECOGNITION Katrin Kirchhoff Department of Electrical Engineering University of Washington , Seattle , WA , USA Dimitra Vergyri SRI International.”

[49] P. Garthwaite and W. Hall, “Easy measures for evaluating non-English corpora for language engineering. Some lessons from Arabic and Bengali.,” 2004.

[50] M. Al-Kabi, H. Wahsheh, I. Alsmadi, E. Al-Shawakfa, a. Wahbeh, and a. Al-Hmoud, “Content-based analysis to detect Arabic web spam,” J. Inf. Sci., vol. 38, no. 3, pp. 284–296, Apr. 2012.

[51] H. A. Wahsheh and M. N. Al-kabi, “Spam Detection Methods for Arabic Web Pages,” pp. 486–490, 2012.

[52] R. Jaramh, T. Saleh, S. Khattab, and I. Farag, “Detecting Arabic Spam Web Pages Using Content Analysis Publication of Little Lion Scientific R & D , Islamabad PAKISTAN,” vol. 6, no. July 2010, 2011.

[53] H. A. Wahsheh and M. N. Al-kabi, “Detecting Arabic Web Spam,” no. 631, pp. 1–8, 2011.

[54] H. a. Wahsheh, M. N. Al-Kabi, and I. M. Alsmadi, “A link and Content Hybrid Approach for Arabic Web Spam Detection,” Int. J. Intell. Syst. Appl., vol. 5, no. 1, pp. 30–43, Dec. 2012.

[55] W. Ben and A. Karaa, “A NEW STEMMER TO I MPROVE I NFORMATION RETRIEVAL,” vol. 5, no. 4, pp. 143–154, 2013.

[56] S. Khoja, “Stemming Arabic Text,” 1999.

[57] K. Taghva, R. Elkhoury, and J. Coombs, “Arabic Stemming Without A Root Dictionary.”

[58] T. Subramaniam, H. A. Jalab, and A. Y. Taqa, “Overview of textual anti-spam filtering techniques,” vol. 5, no. 12, pp. 1869–1882, 2010.

[59] O. Verview, “ITU WSIS T HEMATIC M EETING ON C OUNTERING S PAM C URBING S PAM VIA T ECHNICAL M EASURES : A N.”

[60] W. H. Y. U. S. E. Chrome, “Gmail Intermediate : Increasing Efficiency,” pp. 1–9, 2012.

[61] M. Soranamageswari, “Histogram based Image Spam Detection using Back propagation Neural Networks,” vol. 9, no. 5, 2010.

[62] N. Abhishek, “How Gmail Spam Filter Work ?,” 2010.

[63] B. S. Shaked, “Power User Guide to Gmail,” no. 800, p. 95472, 2012.

[64] L. Vincent, “Google Book Search: Document Understanding on a Massive Scale,” Ninth Int. Conf. Doc. Anal. Recognit. (ICDAR 2007) Vol 2, pp. 819–823, Sep. 2007.

[65] A. Kennedy, “Email Authentication Policy and Deployment Strategy for Financial Services Firms,” 2013.

[66] “How Gmail Blocks Spam,” p. 2007, 2007.

[67] S. E. Pawar, P. P. S. E, and B. M. M, “CAPTCHA : A SECURITY MEASURE AGAINST SPAM ATTACKS,” no. May, pp. 854–857, 2013.

[68] F. Aldrich, “Increase in spam , current controls are insufficient.”

[69] B. Slattery, “Google Explains Gmail ’ s Spam Filtering Process,” 2012.

[70] H. Long, “Windows Live Hotmail Spam Filter Lacking How Do Email Spam Filters Work ? Sender Name,” 2011.

[71] D. Craddock, “We aren ’ t surprised that Hotmail ’ s spam protection is the best in the business,” 2012.

[72] J. E. F. Lists, “Overview of the Junk E-mail Filter,” 2007.

[73] L. Smith, “Outlook 2013 Junk Email Filter,” 2013.

[74] “Deleting Dr . Oz,” 2013.

[75] J. Attenberg, K. Weinberger, A. Dasgupta, A. Smola, and M. Zinkevich, “Collaborative Email-Spam Filtering with the Hashing-Trick,” Conf. Email Anti-Spam, pp. 1–4, 2009.

[76] “Filters in Yahoo Mail ,” no. October, p. 3225, 2013.

[77] “Yahoo mail - spam filter failure,” p. 75.

[78] “Yahoo is really creating a lot of problems,” p. 2012, 2012.

[79] G. Kennedy, “Corpus Linguistics,” pp. 2816–2820, 2001.

[80] S. Alansary, “Building an International Corpus of Arabic (ICA): Progress of Compilation Stage.”

[81] J. Weston and M. Karlen, “Natural Language Processing (Almost) from Scratch,” vol. 12, pp. 2493–2537, 2011.

[82] C. F. Meyer, “English Corpus Linguistics An Introduction.”

[83] K. Simov, G. Popova, P. Osenova, and A. G. B. Str, “HPSG-based syntactic treebank of Bulgarian (BulTreeBank) The BulTreeBank Project Linguistic Modelling Laboratory - CLPPI , Bulgarian Academy of Sciences,” p. 561.

[84] G. Cormack and T. Lynam, “TREC 2005 Spam Track Overview,” no. 2, pp. 1–17, 2005.

[85] B. Klimt and Y. Yang, “Introducing the Enron Corpus.”

[86] G. V Cormack and T. R. Lynam, “TREC 2007 Public Corpus,” p. 2007, 2007.

[87] K. Duh and K. Kirchhoff, “POS Tagging of Dialectal Arabic : A Minimally Supervised Approach.”

[88] V. Telephone and W. Sites, “Communicating with Patients Electronically,” vol. 20001, no. August, 2008.

[89] U. C. Email, U. B. Email, M. Marketing, G. R. Quick, M. M. Fast, M. Python, F. S. One, and D. Rhodes, “What is Spam ? The History of Spam,” 1990.

[90] A. H. Wahbeh and M. Al-kabi, “Comparative Assessment of the Performance of Three WEKA Text Classifiers Applied to Arabic Text,” vol. 21, no. 1, pp. 15–28, 2012.

[91] L. Khreisat, M. Ave, and M. Nj, “Arabic Text Classification Using N-Gram Frequency Statistics A Comparative Study.”

[92] K. C. Ryding, “A Reference Grammar of Modern Standard Arabic,” no. 1998.

[93] S. Sohail, E. Hassanain, and S. Arabia, “Arabic Email Spam Detection Techniques and Related Arabic Text Preprocessing Options : A Survey,” pp. 245–254.

[94] B. R. Bornstein and D. Miao, “Bayesian Spam Filtering,” pp. 1–8, 2011.

[95] P. Graham, “Better Bayesian Filtering,” vol. 2003, no. January, 2003.

[96] I. Androutsopoulos, J. Koutsias, K. V Cbandrinos, and C. D. Spyropoulos, “An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages,” pp. 160–167, 2000.

[97] “C. Elkan, ‘Naïve Bayesian Learning’, Technical Report No. CS97-557,” p. 557.

[98] M. I. Hussien, F. Olayah, M. Al-dwan, and A. Shamsan, “ARABIC TEXT CLASSIFICATION USING SMO , NAÏVE BAYESIAN , J48 ALGORITHMS,” vol. 9, no. November, pp. 306–316, 2011.

[99] “R. M. Duwairi ‘Arabic Text Categorization’, The International Arab Journal of Information Technology , Vol. 4, No.2, April 2007.,” vol. 4, no. 2, p. 2007, 2007.

[100] R. J. Mooney and L. Roy, “Content-Based Book Recommending Using Learning for Text Categorization,” no. June, 2000.

[101] B. Glushko, T. V. Model, T. Weighting, and S. Calculation, “Plan for Today ’ s Class The Boolean Model Boolean Search with Inverted Indexes,” no. November, 2008.

[102] H. Joho and M. Sanderson, “Document frequency and term specificity,” 1972.

[103] T. Task and T. Classifica, “Text Classification and Naïve Bayes.”

[104] G. Bhagyashri, “A UTO E- MAILS C LASSIFICATION U SING B AYESIAN,” vol. 3, no. 4, 2013.

[105] I. Feinerer, K. Hornik, and D. Meyer, “Text mining infrastructure in R,” vol. 25, no. 5, 2008.

[106] S. Manne and S. S. Fatima, “A Novel Approach for Text Categorization of Unorganized data based with Information Extraction,” Int. J. Comput. Sci. Eng., vol. 3, no. 7, pp. 2846–2854, 2011.

[107] M. Feld, M. Kipp, A. Ndiaye, and D. Heckmann, “Weka : Practical machine learning tools and techniques with Java implementations,” 2007.

[108] I. Androutsopoulos, J. Koutsias, K. V Chandrinos, G. Paliouras, and C. D. Spyropoulos, “An Evaluation of Naive Bayesian Anti-Spam Filtering.”

[109] A. Gray and M. Haahr, “Personalised , Collaborative Spam Filtering.”

[110] D. Erickson and N. Mckeown, “The Effectiveness of Whitelisting : a User-Study,” 2007.

[111] J. Jung and E. Sit, “An empirical study of spam traffic and the use of DNS black lists,” Proc. 4th ACM SIGCOMM Conf. Internet Meas. - IMC ’04, p. 370, 2004.

[112] H. Wang, F. Meng, H. Jia, J. Cheng, and J. Xie, “A Keyword Filters Method for Spam via Maximum Independent Sets,” vol. 7, no. 3, pp. 301–310, 2013.

[113] J. Dow, “Suggested Keywords For Spam Filters,” p. 2009, 2009.

[114] D. Zhang, “An Example of Text Classification with Naïve Bayes,” pp. 3–6, 2006.

[115] B. Vidakovic, “Probability , Conditional Probability and Bayes Formula,” pp. 1–13, 2004.

[116] B. G. Robinson, “A Statistical Approach to the Spam Problem,” 2003.

[117] S. Yifrah and G. Lev, “Machine Learning Final Project Spam Email Filtering,” no. March, 2013.

[118] “Stanford Engineering Everywhere CS229 - Machine,” p. 2008, 2008.

[119] M. F. Porter, “The Porter Stemmer Algorithm,” vol. 14, no. 3, p. 1980, 1980.

[120] J. N. Prakash, “Precision and Relative Recall of Search Engines : A Comparative Study of Google and Yahoo,” Singapore Journal of Library & Information Management ,vol. 38, pp. 124–137,2009.

[121] D. M. W. Powers, “Evaluation : From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation,” J. Mach. Learn. Technol., vol. 2, no. 1, pp. 37–63, 2011.

APPENDIXES

Appendix A

A.1 Word Spam

The word spam comes from an old Monty Python Sketch first aired on television in 1970 and written by Michael Palin and Terry Jones in which a two customers are in a greasy café trying to order breakfast from a menu. Spam is in almost every dish on the menu. The sketch shows the woman trying to order something without spam, because she hates it. The word spam for unwanted messages and electronic mail was derived from this sketch. It became so famous for being disgusted and unwanted, that it became a term for junk mail that no one wants, but always get[5]. The first spam ever was a message from a Digital Equipment Corporation (DEC) marketing rep to every Arpanet address on the west coast, or at least the attempt that, they sent it on May 1978.

A.2 First Spam Message

```
Mail-from: DEC-MARLBORO rcvd at 3-May-78 0955-PDT
Date: 1 May 1978 1233-EDT
From: THUERK at DEC-MARLBORO
Subject: ADRIAN@SRI-KL
To: DDAY at SRI-KL, DAY at SRI-KL, DEBOER at UCLA-CCN,
To: WASHDC at SRI-KL, LOGICON at USC-ISI, SDAC at USC-ISI,
To: DELDO at USC-ISI, DELEOT at USC-ISI, DELFINO at USC-ISI,
To: DENICOFF at USC-ISI, DESPAIN at USC-ISI, DEUTSCH at SRI-KL,
To: DEUTSCH at PARC-MAXC, EMY at CCA-TENEX, DIETER at USC-ISIB,
To: DINES at AMES-67, MERADCON at SRI-KL, EPG-SPEC at SRI-KA,
To: DIVELY at SRI-KL, DODD at USC-ISI, DONCHIN at USC-ISIC,
To: JED at LLL-COMP, DORIN at CCA-TENEX, NYU at SRI-KA,
To: DOUGHERTY at USC-ISI, PACOMJ6 at USC-ISI,
To: DEBBY at UCLA-SECURITY, BELL at SRI-KL, JHANNON at SRI-KA,
To: DUBOIS at USC-ISI, DUDA at SRI-KL, POH at USC-ISI,
To: LES at SU-AI, EAST at BBN-TENEX, DEASTMAN at USC-ECL,
To: EBISU at I4-TENEX, NAC at USC-ISIE, ECONOMIDIS at I4-TENEX,
To: WALSH at SRI-KL, GEDWARDS at SRI-KL, WEDWARDS at USC-ISI,
To: NUSC at SRI-KL, RM at SU-AI, ELKIND at PARC-MAXC,
To: ELLENBY at PARC-MAXC, ELLIS at PARC-MAXC, ELLIS at USC-ISIB,
To: ENGELBART at SRI-KL, ENGELMORE at SUMEX-AIM,
To: ENGLISH at PARC-MAXC, ERNST at I4-TENEX,
To: ESTRIN at MIT-MULTICS, EYRES at USC-ISIC,
To: FAGAN at SUMEX-AIM, FALCONER at SRI-KL,
To: DUF at UCLA-SECURITY, FARBER at RAND-UNIX, PMF at SU-AI,
To: HALFF at USC-ISI, RJF at MIT-MC, FEIERBACH at I4-TENEX,
To: FEIGENBAUM at USC-ISI, FEINLER at SRI-KL,
To: FELDMAN at SUMEX-AIM, FELDMAN at SRI-KL, FERNBACH at LLL-COMP,
To: FERRARA at RADC-MULTICS, FERRETTI at SRI-KA,
To: FIALA at PARC-MAXC, FICKAS at USC-ISIC, AFIELD at I4-TENEX,
To: FIKES at PARC-MAXC, REF at SU-AI, FINK at MIT-MULTICS,
To: FINKEL at USC-ISIB, FINN at USC-ISIB, AFGWC at BBN-TENEX,
```

To: FLINT at SRI-KL, WALSH at SRI-KL, DRXAN at SRI-KA,
To: FOX at SRI-KL, FRANCESCHINI at MIT-MULTICS,
To: SAI at USC-ISIC, FREDRICKSON at RAND-RCC, ETAC at BBN-TENEXB,
To: FREYLING at BBN-TENEXE, FRIEDLAND at SUMEX-AIM,
To: FRIENDSHUH at SUMEX-AIM, FRITSCH at LLL-COMP, ME at SU-AI,
To: FURST at BBN-TENEXB, FUSS at LLL-COMP, OP-FYE at USC-ISIB,
To: SCHILL at USC-ISIC, GAGLIARDI at USC-ISIC,
To: GAINES at RAND-UNIX, GALLENSON at USC-ISIB,
To: GAMBLE at BBN-TENEXE, GAMMILL at RAND-UNIX,
To: GANAN at USC-ISI, GARCIA at SUMEX-AIM,
To: GARDNER at SUMEX-AIM, MCCUTCHEN at SRI-KL,
To: GARDNER at MIT-MULTICS, GARLICK at SRI-KL,
To: GARVEY at SRI-KL, GAUTHIER at USC-ISIB,
To: USGS-LIA at BBN-TENEX, GEMOETS at I4-TENEX,
To: GERHART at USC-ISIB, GERLA at USC-ISIE, GERLACH at I4-TENEX,
To: GERMAN at HARV-10, GERPHEIDE at SRI-KA, DANG at SRI-KL,
To: GESCHKE at PARC-MAXC, GIBBONS at CMU-10A,
To: GIFFORD.COMPSYS at MIT-MULTICS, JGILBERT at BBN-TENEXB,
To: SGILBERT at BBN-TENEXB, SDAC at USC-ISI,
To: GILLOGLY at RAND-UNIX, STEVE at RAND-UNIX,
To: GLEASON at SRI-KL, JAG;BIN(1525) at UCLA-CCN,
To: GOLD at LL-11, GOLDBERG at USC-ISIB, GOLDGERG at SRI-KL,
To: GROBSTEIN at SRI-KL, GOLDSTEIN at BBN-TENEXB,
To: DARPM-NW at BBN-TENEXB, GOODENOUGH at USC-ISIB,
To: GEOFF at SRI-KL, GOODRICH at I4-TENEX, GOODWIN at USC-ISI,
To: GOVINSKY at SRI-KL, DEAN at I4-TENEX, TEG at MIT-MULTICS,
To: CCG at SU-AI, EPG-SPEC at SRI-KA, GRISS at USC-ECL,
To: BJG at RAND-UNIX, MCCUTCHEN at SRI-KL, GROBSTEIN at SRI-KL,
To: MOBAH at I4-TENEX, GUSTAFSON at USC-ISIB, GUTHARY at SRI-KL,
To: GUTTAG at USC-ISIB, GUYTON at RAND-RCC,
To: ETAC-AD at BBN-TENEXB, HAGMANN at USC-ECL, HALE at I4-TENEX,
To: HALFF at USC-ISI, DEHALL at MIT-MULTICS,
To: HAMPEL at LLL-COMP, HANNAH at USC-ISI,
To: NORSAR-TIP at USC-ISIC, SCRL at USC-ISI, HAPPY at SRI-KL,
To: HARDY at SRI-KL, IMPACT at SRI-KL, KLH at SRI-KL,
To: J33PAC at USC-ISI, HARRISON at SRI-KL, WALSH at SRI-KL,
To: DRCPM-FF at BBN-TENEXB, HART at AMES-67, HART at SRI-KL,
To: HATHAWAY at AMES-67, AFWL at I4-TENEX, BHR at RAND-UNIX,
To: RICK at RAND-UNIX, DEBE at USC-ISIB, HEARN at USC-ECL,
To: HEATH at UCLA-ATS, HEITMEYER at BBN-TENEX, ADTA at SRI-KA,
To: HENDRIX at SRI-KL, CH47M at BBN-TENEXB, HILLIER at SRI-KL,
To: HISS at I4-TENEX, ASLAB at USC-ISIC, HOLG at USC-ISIB,
To: HOLLINGWORTH at USC-ISIB, HOLLOWAY at HARV-10,
To: HOLMES at SRI-KL, HOLSWORTH at SRI-KA, HOLT at LLL-COMP,
To: HOLTHAM at LL, DHOLZMAN at RAND-UNIX, HOPPER at USC-ISIC,
To: HOROWITZ at USC-ISIB, VSC at USC-ISI, HOWARD at LLL-COMP,
To: HOWARD at USC-ISI, PURDUE at USC-ISI, HUBER at RAND-RCC,
To: HUNER at RADC-MULTICS, HUTSON at AMES-67, IMUS at USC-ISI,
To: JACOBS at USC-ISIE, JACOBS at BBN-TENEXB,
To: JACQUES at BBN-TENEXB, JARVIS at PARC-MAXC,
To: JEFFERS at PARC-MAXC, JENKINS at PARC-MAXC,
To: JENSEN at SRI-KA, JIRAK at SUMEX-AIM, NICKIE at SRI-KL,

To: JOHNSON at SUMEX-AIM, JONES at SRI-KL, JONES at LLL-COMP,
To: JONES at I4-TENEX, RLJ at MIT-MC, JURAK at USC-ECL,
To: KAHLER at SUMEX-AIM, MWK at SU-AI, KAINE at USC-ISIB,
To: KALTGRAD at UCLA-ATS, MARK at UCLA-SECURITY, RAK at SU-AI,
To: KASTNER at USC-ISIB, KATT at USC-ISIB,
To: UCLA-MNC at USC-ISI, ALAN at PARC-MAXC, KEENAN at USC-ISI,
To: KEHL at UCLA-CCN, KELLEY at SRI-KL, BANANA at I4-TENEX,
To: KELLOGG at USC-ISI, DDI at USC-ISI, KEMERY at SRI-KL,
To: KEMMERER at UCLA-ATS, PARVIZ at UCLA-ATS, KING at SUMEX-AIM,
To: KIRSTEIN at USC-ISI, SDC at UCLA-SECURITY,
To: KLEINROCK at USC-ISI, KLEMBA at SRI-KL, CSK at USC-ISI,
To: KNIGHT at SRI-KL, KNOX at USC-ISI, KODA at USC-ISIB,
To: KODANI at AMES-67, KOOIJ at USC-ISI, KREMERS at SRI-KL,
To: BELL at SRI-KL, KUNZELMAN at SRI-KL, PROJX at SRI-KL,
To: LAMPSON at PARC-MAXC, SDL at RAND-UNIX, JOJO at SRI-KL,
To: SDC at USC-ISI, NELC3030 at USC-ISI,
To: LEDERBERG at SUMEX-AIM, LEDUC at SRI-KL, JSLEE at USC-ECL,
To: JACOBS at USC-ISIE, WREN at USC-ISIB, LEMONS at USC-ISIB,
To: LEUNG at SRI-KL, J33PAC at USC-ISI, LEVIN at USC-ISIB,
To: LEVINTHAL at SUMEX-AIM, LICHTENBERGER at I4-TENEX,
To: LICHTENSTEIN at USC-ISI, LIDDLE at PARC-MAXC,
To: LIEB at USC-ISIB, LIEBERMAN at SRI-KL, STANL at USC-ISIE,
To: LIERE at I4-TENEX, DOCB at USC-ISIC, LINDSAY at SRI-KL,
To: LINEBARGER at AMES-67, LIPKIS at USC-ECL, SLES at USC-ISI,
To: LIS at SRI-KL, LONDON at USC-ISIB, J33PAC at USC-ISI,
To: LOPER at SRI-KA, LOUVIGNY at SRI-KL, LOVELACE at USC-ISIB,
To: LUCANIC at SRI-KL, LUCAS at USC-ISIB, DCL at SU-AI,
To: LUDLAM at UCLA-CCN, YNGVAR at SRI-KA, LYNCH at SRI-KL,
To: LYNN at USC-ISIB, MABREY at SRI-KL, MACKAY at AMES-67,
To: MADER at USC-ISIB, MAGILL at SRI-KL, KMAHONEY at BBN-TENEX,
To: MANN at USC-ISIB, ZM at SU-AI, MANNING at USC-ISI,
To: MANTIPLY at I4-TENEX, MARIN at I4-TENEX, SCRL at USC-ISI,
To: HARALD at SRI-KA, GLORIA-JEAN at UCLA-CCN, MARTIN at USC-ISIC,
To: WMARTIN at USC-ISI, GRM at RAND-UNIX, MASINTER at USC-ISI,
To: MASON at USC-ISIB, MATHIS at SRI-KL, MAYNARD at USC-ISIC,
To: MCBREARTY at SRI-KL, MCCALL at SRI-KA, MCCARTHY at SU-AI,
To: MCCLELLAND at USC-ISI, DORIS at RAND-UNIX, MCCLURG at SRI-KL,
To: JOHN at I4-TENEX, MCCREIGHT at PARC-MAXC, MCCRUMB at USC-ISI,
To: DRXTE at SRI-KA
cc: BPM at SU-AI

Appendix B

B.1 Suggested English Keywords for Spam Filters

Free!, 50% off!, Click Here, Call now!, Subscribe, Earn $, Discount!, Eliminate Debt, enlargement, Double your income, You're a Winner!, Reverses Aging, "Hidden", Information you requested, "Stop" or "Stops", Lose Weight, medication, Multi level Marketing, Million Dollars, Opportunity, Compare, Removes, Collect Amazing, Cash Bonus, Promise You, Credit, Loans, Satisfaction Guaranteed, Serious Cash, Search Engine Listings, Act Now!, All New, All Natural, Avoid Bankruptcy, As Seen On..., Buy Direct, Casino, Cash, Consolidate Your Debt, Special Promotion, Easy Terms, Get Paid, Guarantee, Guaranteed, Great offer, Give it away, Giving it away, Join millions, Meet Singles, MLM, mortgage, No cost, No fees, Offer, One time, Online pharmacy, Online marketing, Order Now, Please Read, Don't Delete, Save up to, Time limited, Unsecured debt or credit, Vacation, Viagra, Visit our web site, While Supplies last, Why pay more?, Winner, Work at home, You've been selected.

B.2 Arabic Spam Keywords for Spam Filters

Table B.1 Arabic spam words for spam filters

طبيعي100%	مكاسب نقدية إضافية	كلمة السر	التمويل	مرحبا	قريبا
عضوية حره	زيارة موقعنا على الانترنت	الارباح المحتملة	رخيص	اتصل	انتبه
الوصول الحر	الاعمال التجارية على الانترنت	50% اقل	استثمار	الانترنت	ائتمان
هدية مجانية	درجة علمية على الانترنت	الدخل الاضافي	أسهم	صديق	مشتريات
النقد السريع	الربح الخالص	100% مجاني	دخلك	افتح	قرار
اخسر وزنك	اسعار معقولة	إنضم للملايين	البطاقة	عروض	تخفيضات
السعر الأقل	الكسب خلال اسبوع	ضاعف دخلك	اكسب	اربح	تسوق
عضوية حره	قارن بين الاسعار	صديقي العزيز	تجارة	الفوز	زيارة
صفقة رائعة	مكاتب الائتمان	السعر الأفضل	حلول	فرصة	مال
لا تتردد	تخفيضات تصل الى 50%	سعر الاسهم	قبول	عطلة	مليون
الدخل المنزل	البطاقات المقبولة	كسب المال السريع	حرية	تجريبي	نجاح
انها فعالة	بطاقات الائتمانية	اعادة التمويل	معجزة	مجانا	شريك
سن المراهقة	الأعمال التجارية المنزلية	التوظيف المنزلي	حياة	معتمد	معدل
الوصول الحر	قارن بين الأسعار	التحميل المجاني	ضمان	مئه	تحميل
عرض رائع	الاشتراك مجانا اليوم	التثبيت المجاني	عاجل	محدود	فوريه
منحه ماليه	الالغاء فى اى وقت	إنخفاض شديد	شهادة	تهنئه	مدهش
عينة مجانية	خالية من المخاطر	استشارة مجانية	وصول	علاوة	شراء
البريد المجاني	إستضافة مجانية	التركيب مجاني	مبيعات	كلفة	بالمجان
	مره واحدة في العمر	أسعار مزهلة	فقط	طلبات	اعدكم
	إحصل عليه الآن	تنتهى العروض	اداء	احفظ	قرض
	العلامة التجارية الجديدة	برامج مجانية	فرصة	صفقة	عروض
			الف	نجاح	تثبيت

Appendix C

C.1 Personalized Spam Detection Email Code

Personalized Spam Detection Email Home page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Home Page</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="oneColLiqCtrHdrhome.css" rel="stylesheet" type="text/css" />
<style type="text/css">
.container.content h3 {
        font-family: Verdana, Geneva, sans-serif;
}
.container.content p {
        font-family: Verdana, Geneva, sans-serif;
}
.container.content p {
        font-family: Times New Roman, Times, serif;
}
.container.content p {
        font-family: Courier New, Courier, monospace;
}
.container.content.promo a strong samp {
        font-family: Verdana, Geneva, sans-serif;
}
.container.content.promo a strong samp {
        font-family: Georgia, Times New Roman, Times, serif;
}

.container.content h3 {
        font-family: Verdana, Geneva, sans-serif;
}
.container.content p {
        font-family: Verdana, Geneva, sans-serif;
}
.container.content p {
        font-family: Times New Roman, Times, serif;
}
.container.content p {
        font-family: Courier New, Courier, monospace;
}
.container.content.promo a strong samp {
        font-family: Verdana, Geneva, sans-serif;
}
.container.content.promo a strong samp {
        font-family: Georgia, Times New Roman, Times, serif;
```

```
}
</style>
</head>
<body>
<div class="container">
 <div class="header">
 <h1 align="center"><span class="promo">Personalized Spam Detection
Email</span></h1>
 <!-- end.header --></div>
 <div class="content">
 <h1 align="center"> </h1>
 <h1 align="center" class="promo"><a href="login.asp"><strong><samp>User
Login</samp></strong></a></h1>
 <p align="center"> </p>
 <p><strong>welcome to your family mail</strong></p>
 </div>
 <div class="footer">
 <div align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font> ©
</div>
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
Create New User
<%@LANGUAGE="VBSCRIPT" CODEPAGE="65001"%>
<!--#include file="Connections/emailcn.asp" -->
<%
' *** Validate request to log in to this site.
MM_LoginAction = Request.ServerVariables("URL")
If Request.QueryString <> "" Then MM_LoginAction = MM_LoginAction + "?" +
Server.HTMLEncode(Request.QueryString)
MM_valUsername = CStr(Request.Form("Login"))
If MM_valUsername <> "" Then
 Dim MM_fldUserAuthorization
 Dim MM_redirectLoginSuccess
 Dim MM_redirectLoginFailed
 Dim MM_loginSQL
 Dim MM_rsUser
 Dim MM_rsUser_cmd
 MM_fldUserAuthorization = ""
 MM_redirectLoginSuccess = "inbox.asp"
 MM_redirectLoginFailed = "login.asp"
 MM_loginSQL = "SELECT login, pwd"
 If MM_fldUserAuthorization <> "" Then MM_loginSQL = MM_loginSQL & "," &
MM_fldUserAuthorization
```

```
 MM_loginSQL = MM_loginSQL & " FROM login WHERE login =? AND pwd =?"
 Set MM_rsUser_cmd = Server.CreateObject ("ADODB.Command")
 MM_rsUser_cmd.ActiveConnection = MM_emailcn_STRING
 MM_rsUser_cmd.CommandText = MM_loginSQL
 MM_rsUser_cmd.Parameters.Append MM_rsUser_cmd.CreateParameter("param1",
200, 1, 255, MM_valUsername) ' adVarChar
 MM_rsUser_cmd.Parameters.Append MM_rsUser_cmd.CreateParameter("param2",
200, 1, 255, Request.Form("pwd")) ' adVarChar
 MM_rsUser_cmd.Prepared = true
 Set MM_rsUser = MM_rsUser_cmd.Execute
 If Not MM_rsUser.EOF Or Not MM_rsUser.BOF Then
 ' username and password match - this is a valid user
 Session("MM_Username") = MM_valUsername
 If (MM_fldUserAuthorization <> "") Then
 Session("MM_UserAuthorization") =
CStr(MM_rsUser.Fields.Item(MM_fldUserAuthorization).Value)
 Else
 Session("MM_UserAuthorization") = ""
 End If
 if CStr(Request.QueryString("accessdenied")) <> "" And false Then
 MM_redirectLoginSuccess = Request.QueryString("accessdenied")
 End If
 MM_rsUser.Close
 Response.Redirect(MM_redirectLoginSuccess)
 End If
 MM_rsUser.Close
 Response.Redirect(MM_redirectLoginFailed)
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Home Page</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="oneColLiqCtrHdrhome.css" rel="stylesheet" type="text/css" />
<style type="text/css">
.container.content h3 {
	font-family: Verdana, Geneva, sans-serif;
}
.container.content form div table tr td {
	font-family: Tahoma, Geneva, sans-serif;
}
.container.content form div table tr td {
	font-family: Georgia, Times New Roman, Times, serif;
```

```
}
.container.content form div table tr td {
        font-family: Georgia, Times New Roman, Times, serif;
</style>
</head>
<body>
<div class="container">
 <div class="header"><!-- end.header -->
 <h1 align="center"><span class="promo">Personalized Spam Detection
Email</span></h1>
 </div>
 <div class="content">
 <h2 align="center">New User</h2>
 <p><strong>Please enter all the Information on the form
 </strong></p>
 <table width="361" border="1" bgclor="#99CC99">
 <tr>
 <td width="162" bgcolor="#669999"><strong>Login</strong></td>
 <td width="183" bgcolor="#669999"><strong>
 <input type="text" name="uid" id="uid2" />
 </strong></td>
 </tr>
 <tr>
 <td bgcolor="#669999"><strong>Password</strong></td>
 <td bgcolor="#669999"><strong>
 <input type="password" name="uid2" id="uid2" />
 </strong></td>
 </tr>
 <tr>
 <td bgcolor="#669999"><strong>Renter Password</strong></td>
 <td bgcolor="#669999"><strong>
 <input type="password" name="uid9" id="uid9" />
 </strong></td>
 </tr>
 <tr>
 <td bgcolor="#669999"><strong>First Name</strong></td>
 <td bgcolor="#669999"><strong>
 <input type="text" name="fnid" id="fnid" />
 </strong></td>
 </tr>
 <tr>
 <td bgcolor="#669999"><strong>Last Name</strong></td>
 <td bgcolor="#669999"><strong>
 <input type="text" name="lnid" id="lnid" />
 </strong></td>
 </tr>
```

```
<tr>
<td bgcolor="#669999"><strong>Job</strong></td>
<td bgcolor="#669999"><strong>
<select name="jid" id="jid">
<option>Student</option>
<option>Teacher</option>
<option>Employee</option>
</select>
</strong></td>
</tr>
<tr>
<td bgcolor="#669999"><strong>Date of Birth</strong></td>
<td bgcolor="#669999"><strong>
<input type="text" name="uid6" id="uid6" />
</strong></td>
</tr>
<tr>
<td bgcolor="#669999"><strong>Gender</strong></td>
<td bgcolor="#669999"><strong>
<input type="radio" name="radio" id="mid" value="mid" />
<label for="mid">Male
<input type="radio" name="radio" id="mid2" value="fid" />
Female</label>
</strong></td>
</tr>
<tr>
<td bgcolor="#669999"><strong>Country</strong></td>
<td bgcolor="#669999"><strong>
<select name="cid" size="1" id="cid">
<option>Egypt</option>
<option>Oman</option>
<option>Saudi</option>
<option>Kwait</option>
<option>Sudan</option>
</select>
</strong></td>
</tr>
<tr>
<td bgcolor="#669999"><input type="submit" name="sid" id="sid" value="Send"
/></td>
<td bgcolor="#669999"> </td>
</tr> </table>
<div align="center"> </div>
</p>
```

```
 <div align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font> ©
</div>
 </div><div class="footer">
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
User Login
<%@LANGUAGE="VBSCRIPT" CODEPAGE="65001"%>
<!--#include file="Connections/emailcn.asp" -->
<%
' *** Validate request to log in to this site.
MM_LoginAction = Request.ServerVariables("URL")
If Request.QueryString <> "" Then MM_LoginAction = MM_LoginAction + "?" +
Server.HTMLEncode(Request.QueryString)
MM_valUsername = CStr(Request.Form("Login"))
If MM_valUsername <> "" Then
 Dim MM_fldUserAuthorization
 Dim MM_redirectLoginSuccess
 Dim MM_redirectLoginFailed
 Dim MM_loginSQL
 Dim MM_rsUser
 Dim MM_rsUser_cmd
 MM_fldUserAuthorization = ""
 MM_redirectLoginSuccess = "inbox.asp"
 MM_redirectLoginFailed = "login.asp"
 MM_loginSQL = "SELECT login, pwd"
 If MM_fldUserAuthorization <> "" Then MM_loginSQL = MM_loginSQL & "," &
MM_fldUserAuthorization
 MM_loginSQL = MM_loginSQL & " FROM login WHERE login =? AND pwd =?"
 Set MM_rsUser_cmd = Server.CreateObject ("ADODB.Command")
 MM_rsUser_cmd.ActiveConnection = MM_emailcn_STRING
 MM_rsUser_cmd.CommandText = MM_loginSQL
 MM_rsUser_cmd.Parameters.Append MM_rsUser_cmd.CreateParameter("param1",
200, 1, 255, MM_valUsername) ' adVarChar
 MM_rsUser_cmd.Parameters.Append MM_rsUser_cmd.CreateParameter("param2",
200, 1, 255, Request.Form("pwd")) ' adVarChar
 MM_rsUser_cmd.Prepared = true
 Set MM_rsUser = MM_rsUser_cmd.Execute
 If Not MM_rsUser.EOF Or Not MM_rsUser.BOF Then
 ' username and password match - this is a valid user
 Session("MM_Username") = MM_valUsername
 If (MM_fldUserAuthorization <> "") Then
 Session("MM_UserAuthorization") =
CStr(MM_rsUser.Fields.Item(MM_fldUserAuthorization).Value)
```

```
 Else
 Session("MM_UserAuthorization") = ""
 End If
 if CStr(Request.QueryString("accessdenied")) <> "" And false Then
 MM_redirectLoginSuccess = Request.QueryString("accessdenied")
 End If
 MM_rsUser.Close
 Response.Redirect(MM_redirectLoginSuccess)
 End If
 MM_rsUser.Close
 Response.Redirect(MM_redirectLoginFailed)
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Home Page</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="oneColLiqCtrHdrhome.css" rel="stylesheet" type="text/css" />
<style type="text/css">
.container.content h3 {
        font-family: Verdana, Geneva, sans-serif;
}
.container.content form div table tr td {
        font-family: Tahoma, Geneva, sans-serif;
}
.container.content form div table tr td {
        font-family: Georgia, Times New Roman, Times, serif;
}
.container.content form div table tr td {
        font-family: Georgia, Times New Roman, Times, serif;
</style>
</head>
<body>
<div class="container">
 <div class="header"><!-- end.header -->
 <h1 align="center"><span class="promo">Personalized Spam Detection
Email</span></h1>
 </div>
 <div class="content">
 <h1 align="center"> </h1>
 <form ACTION="<%=MM_LoginAction%>" METHOD="POST" name="loginfr">
 <div align="center">
 <table width="413" border="0">
```

```
 <tr>
    <td width="94" bgcolor="#99CC99"><strong>Login</strong></td>
 <td width="309" bgcolor="#99CC99"><input name="Login" type="text" id="Login"
size="18" />
 @<strong>permail.com</strong></td>
 </tr>
   <tr>
  <td width="94" bgcolor="#99CC99"><strong>Password</strong></td>
 <td bgcolor="#99CC99"><input name="pwd" type="password" id="pwd" size="18"
/></td>
 </tr>
   <tr>
  <td bgcolor="#99CC99"><input type="submit" name="log" id="log" value="Login"
/></td>
 <td bgcolor="#99CC99"><input type="button" name="button" id="button" value="New
User" /></td>
 </tr>
   </table>
 </div>
 </form>
 <div align="center"> </div>
 </p>
 <div align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font> ©
</div>
 </div><div class="footer">
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
```

User Inbox

```
<%@LANGUAGE="VBSCRIPT"%>
<%
' *** Logout the current user.
MM_Logout = CStr(Request.ServerVariables("URL")) & "?MM_Logoutnow=1"
If (CStr(Request("MM_Logoutnow")) = "1") Then
 Session.Contents.Remove("MM_Username")
 Session.Contents.Remove("MM_UserAuthorization")
 MM_logoutRedirectPage = "index.htm"
 ' redirect with URL parameters (remove the "MM_Logoutnow" query param).
 if (MM_logoutRedirectPage = "") Then MM_logoutRedirectPage =
CStr(Request.ServerVariables("URL"))
 If (InStr(1, UC_redirectPage, "?", vbTextCompare) = 0 And Request.QueryString <> "")
Then
 MM_newQS = "?"
 For Each Item In Request.QueryString
```

```
 If (Item <> "MM_Logoutnow") Then
 If (Len(MM_newQS) > 1) Then MM_newQS = MM_newQS & "&"
 MM_newQS = MM_newQS & Item & "=" &
Server.URLencode(Request.QueryString(Item))
 End If
 Next
 if (Len(MM_newQS) > 1) Then MM_logoutRedirectPage = MM_logoutRedirectPage &
MM_newQS
 End If
 Response.Redirect(MM_logoutRedirectPage)
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>User Inbox</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="twoColLiqLtHdrinbx.css" rel="stylesheet" type="text/css" /><!--[if lte IE
7]>
<style>
.content { margin-right: -1px; } /* this 1px negative margin can be placed on any of the
columns in this layout with the same corrective effect. */
ul.nav a { zoom: 1; } /* the zoom property gives IE the hasLayout trigger it needs to
correct extra whiltespace between the links */
</style>
<![endif]-->
<style type="text/css">
.container.sidebar1.nav li a {
        font-family: Verdana, Geneva, sans-serif;
}
.container.sidebar1.nav li a {
        font-family: Georgia, Times New Roman, Times, serif;
}
</style>
</head>
<body>
<div class="container">
 <div class="header"><!-- end.header -->
 <div align="center">
 <h3><strong> Personalized Spam Detection Email </strong></h3>
 </div>
 </div>
 <div class="sidebar1">
 <ul class="nav">
```

```
 <li><a href="login.asp">login by other user</a></li>
 <li><a href="junk.asp">Junk Mail</a></li>
 <li><a href="whitelist.asp">Whitelist</a></li>
 <li><a href="blacklist.asp">Blacklist</a></li>
 <li><a href="<%= MM_Logout %>">Logout</a></li>
 </ul>
 <!-- end.sidebar1 --></div>
 <div class="content">
 <h2>User Inbox</h2>
 <p>
<%
 set email=server.CreateObject("ADODB.Connection")
        set rs=server.CreateObject("ADODB.RecordSet")
        set rs1=server.CreateObject("ADODB.RecordSet")
        set rslog=server.CreateObject("ADODB.RecordSet")
        dim sqlstr
        dim sqlbl
        dim sqlwl
        logsess = Session("MM_Username")
        response.write("<font color=red><b>"& logsess &"@permail.com")
        mbox=Request("mbox")
        response.write(""& mbox &"")
        nspm=Request("nspm")
        response.write(""& nspm &"")
        page=request("page")
 email.open "permail"
 rslog.open "Select * from login where login= '"& logsess &"'",email
 user = "" & rslog(0) &""
        rs.open "SELECT * FROM message INNER JOIN usermsg ON
message.msid=usermsg.msid WHERE usermsg.uid = "& user &"",email
 response.write("<p align=left>")
 on error resume next
        rs.pagesize= 22
        response.write("<p align=left><tap align=left><form name=msgfr
action=junk.asp method= post>")
        response.write("<p align=left><table align=center cellspacing=2 cellpadding=0
border=0 >")
        Response.Write("<font color=#000000><tr bgcolor=#99CC99><th
align=center>No<th align=center>Delete Message<th>Is Spam<th>From<th
align=center>Date<th>Subject")
        if len(page)>0 then
                rs.absolutepage=page
        else
                rs.absolutepage=1
        end if
        bgflip=true
```

```
        for i=1 to rs.pagesize
        if not rs.EOF then
                if bgflip then
                        Response.Write("<tr bgcolor=#eeeeee>")
                else
                        Response.Write("<tr bgcolor=#eeeeff>")
                end if
                bgflip=not bgflip
 Response.Write("<td align=center><li><td align=center><a href=msgdel.asp?msgid="
& rs(0) & "><font size=4 face=Times New Roman color = blue> Delete <td
align=center><a href=msgspm.asp?msgid=" & rs(0) & "><font face=Times New Roman
size=4 color=red>Spam<td align=left ><font face=Times New Roman size=4>" & rs(1)
& " <td align=left ><font face=Times New Roman size=4>" & rs(3) & " <td
align=left ><font face=Times New Roman size=4><a href=msgcont.asp?msgid=" &
rs(0) & ">" & rs(4) & " <td align=left >")
                rs.MoveNext
        end if
        next
        response.write("</table>")
        response.write("</table>")
        response.write("</form>")
        rs.Close
        rs1.Close
        email.Close
        set rs=nothing
        set rs1=nothing
        set email=nothing
        %>
 <!-- end.content --></div>
 <div class="footer">
 <p align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font>
©</p>
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
```

User Junk Mail

```
<%@LANGUAGE="VBSCRIPT"%>
<%
' *** Logout the current user.
MM_Logout = CStr(Request.ServerVariables("URL")) & "?MM_Logoutnow=1"
If (CStr(Request("MM_Logoutnow")) = "1") Then
 Session.Contents.Remove("MM_Username")
 Session.Contents.Remove("MM_UserAuthorization")
 MM_logoutRedirectPage = "index.htm"
```

```
 ' redirect with URL parameters (remove the "MM_Logoutnow" query param).
 if (MM_logoutRedirectPage = "") Then MM_logoutRedirectPage =
CStr(Request.ServerVariables("URL"))
 If (InStr(1, UC_redirectPage, "?", vbTextCompare) = 0 And Request.QueryString <> "")
Then
 MM_newQS = "?"
 For Each Item In Request.QueryString
 If (Item <> "MM_Logoutnow") Then
 If (Len(MM_newQS) > 1) Then MM_newQS = MM_newQS & "&"
 MM_newQS = MM_newQS & Item & "=" &
Server.URLencode(Request.QueryString(Item))
 End If
 Next
 if (Len(MM_newQS) > 1) Then MM_logoutRedirectPage = MM_logoutRedirectPage &
MM_newQS
 End If
 Response.Redirect(MM_logoutRedirectPage)
End If
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Junk Mail</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="twoColLiqLtHdrinbx.css" rel="stylesheet" type="text/css" /><!--[if lte IE
7]>
<style>
.content { margin-right: -1px; } /* this 1px negative margin can be placed on any of the
columns in this layout with the same corrective effect. */
ul.nav a { zoom: 1; } /* the zoom property gives IE the hasLayout trigger it needs to
correct extra whiltespace between the links */
</style>
<![endif]-->
<style type="text/css">
.container.sidebar1.nav li a {
        font-family: Georgia, Times New Roman, Times, serif;
}
</style>
</head>
<body>
<div class="container">
 <div class="header">
 <div align="center"><strong>Personalized Spam Detection Email </strong></div>
 <!-- end.header --></div>
```

```
<div class="sidebar1">
<ul class="nav">
<li><a href="login.asp">login by other user</a></li>
<li><a href="inbox.asp">Inbox</a></li>
<li><a href="whitelist.asp">Whitelist</a></li>
<li><a href="blacklist.asp">Blacklist</a></li>
<li><a href="<%= MM_Logout %>">Logout</a></li>
</ul>
<!-- end.sidebar1 --></div>
<div class="content">
<h2>Junk Mail</h2>
<p>
<%
set email=server.CreateObject("ADODB.Connection")
        set rs=server.CreateObject("ADODB.RecordSet")
        set rs1=server.CreateObject("ADODB.RecordSet")
        set rslog=server.CreateObject("ADODB.RecordSet")
        dim sqlstr
        dim sqlbl
        dim sqlwl
        logsess = Session("MM_Username")
        response.write("<font color=red><b>"& logsess &"@permail.com")
        mbox=Request("mbox")
        response.write(""& mbox &"")
        nspm=Request("nspm")
        response.write(""& nspm &"")
        page=request("page")
email.open "permail"
rslog.open "Select * from login where login= '"& logsess &"'",email
user = "" & rslog(0) &""
        rs.open "SELECT * FROM junk INNER JOIN userjn ON junk.jid=userjn.jid
WHERE userjn.uid = "& user &"",email
response.write("<p align=left>")
on error resume next
        rs.pagesize=15
        response.write("<p align=left><tap align=left><form name=msgfr
action=junk.asp method= post>")
        response.write("<p align=left><table align=center cellspacing=2 cellpadding=0
border=0 >")
        Response.Write("<font color=#000000><tr bgcolor=#99CC99><th
align=center>No<th align=center>Move to Inbox<th>Not Spam<th>From<th
align=center>Date<th>Subject")
        if len(page)>0 then
                rs.absolutepage=page
        else
                rs.absolutepage=1
```

```
        end if
        bgflip=true
        for i=1 to rs.pagesize
        if not rs.EOF then
                if bgflip then
                        Response.Write("<tr bgcolor=#eeeeee>")
                else
                        Response.Write("<tr bgcolor=#eeeeff>")
                end if
                bgflip=not bgflip
Response.Write("<td align=left><li><td align=center><font face=Times New Roman
size=4><a href=jnkmbox.asp?msgid=" & rs(0) & "><font color=#cc00ff> To Inbox<td
align=left><a href=jnknspm.asp?msgid=" & rs(0) & "><font face=Times New Roman
size=4 color=#ff6666>Not Spam<td align=left ><font face=Times New Roman size=4>"
& rs(1) & " <td align=left >" & rs(3) & " <td align=left ><font face=Times New Roman
size=4><a href=jnkcont.asp?msgid=" & rs(0) & ">" & rs(4) & " <td align=left >")
                rs.MoveNext
        end if
        next
        response.write("</table>")
        response.write("</table>")
        response.write("</form>")
        '------------------------ Action ----------------------------------
        rs.Close
        rs1.Close
        email.Close
        set rs=nothing
        set rs1=nothing
        set email=nothing
        %>
  </p>

 <p> </p>
 <!-- end.content --></div>
 <div class="footer">
 <p align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font>
©</p>
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
User Whitelist
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>User Withelist</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="twoColLiqLtHdrinbx.css" rel="stylesheet" type="text/css" /><!--[if lte IE
7]>
<style>
.content { margin-right: -1px; } /* this 1px negative margin can be placed on any of the
columns in this layout with the same corrective effect. */
ul.nav a { zoom: 1; } /* the zoom property gives IE the hasLayout trigger it needs to
correct extra whiltespace between the links */
</style>
<![endif]-->
<style type="text/css">
.container.sidebar1.nav li a {
        font-family: Georgia, Times New Roman, Times, serif;
}
</style>
</head>
<body>
<div class="container">
 <div class="header">
 <div align="center"><strong>Personalized Spam Detection Email </strong></div>
 <!-- end.header --></div>
 <div class="sidebar1">
 <ul class="nav">
 <li><a href="login.asp">login by other user</a></li>
 <li><a href="inbox.asp">Inbox</a></li>
 <li><a href="junk.asp">Junk Mail</a></li>
 <li><a href="blacklist.asp">Blacklist</a></li>
 <li class="promo"><a href="vocablist.asp">Vocabulary list</a></li>
 <li><a href="<%= MM_Logout %>">Logout</a></li>
 </ul>
 </div>
 <h2>User Whitelist Address</h2>
 <p>
<%
 set email=server.CreateObject("ADODB.Connection")
        set rs=server.CreateObject("ADODB.RecordSet")
        set rs1=server.CreateObject("ADODB.RecordSet")
        set rslog=server.CreateObject("ADODB.RecordSet")
        dim sqlstr
        dim sqlbl
        dim sqlwl
        logsess = Session("MM_Username")
        response.write("<font color=red><b>"& logsess &"@permail.com")
```

```
        mbox=Request("mbox")
        response.write(""& mbox &"")
        nspm=Request("nspm")
        response.write(""& nspm &"")
 page=request("page")
email.open "permail"
 rslog.open "Select * from login where login= '"& logsess &"'",email
 user = "" & rslog(0) &""
        rs.open "SELECT whitelist.wlid, whitelist.wladd FROM whitelist INNER JOIN
(login INNER JOIN userwl ON login.uid = userwl.uid) ON whitelist.wlid = userwl.wlid
where login.uid= "& user &"",email
 response.write("<p align=left>")
 on error resume next
        rs.pagesize=15
        response.write("<p align=left><table align=center cellspacing=2 cellpadding=0
border=0 >")
        Response.Write("<font color=#000000><tr bgcolor=#99CC99><th
align=center>No</th><th align=center>Whitelist Address")
        if len(page)>0 then
                rs.absolutepage=page
        else
                rs.absolutepage=1
        end if
        bgflip=true
        for i=1 to rs.pagesize
        if not rs.EOF then
                if bgflip then
                        Response.Write("<tr bgcolor=#eeeeee>")
                else
                        Response.Write("<tr bgcolor=#eeeeff>")
                end if
                bgflip=not bgflip
 Response.Write("<td align=center><li></td><td align=left><font face=Times New
Roman size=4>" & rs(1) & " ")
                rs.MoveNext
        end if
        next
        response.write("</table>")
        rs.Close
        rs1.Close
        email.Close
        set rs=nothing
        set rs1=nothing
        set email=nothing
        %>
  </p>
```

```
 <p> </p>
 <!-- end.content --></div>
 <div class="footer">
 <p align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font>
©</p>
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
```

User Blacklist

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>User Blacklist</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="twoColLiqLtHdrinbx.css" rel="stylesheet" type="text/css" /><!--[if lte IE
7]>
<style>
.content { margin-right: -1px; } /* this 1px negative margin can be placed on any of the
columns in this layout with the same corrective effect. */
ul.nav a { zoom: 1; } /* the zoom property gives IE the hasLayout trigger it needs to
correct extra whiltespace between the links */
</style>
<![endif]-->
<style type="text/css">
.container.sidebar1.nav li a {
        font-family: Georgia, Times New Roman, Times, serif;
}
</style>
</head>
<body bgcolor="#333333">
<div class="container">
 <div class="header">
 <div align="center"><strong>Personalized Spam Detection Email </strong></div>
 <!-- end.header --></div>
 <div class="sidebar1">
 <ul class="nav">
 <li><a href="login.asp">login by other user</a></li>
 <li><a href="inbox.asp">Inbox</a></li>
 <li><a href="junk.asp">Junk Mail</a></li>
 <li><a href="whitelist.asp">Whitelist</a></li>
 <li class="promo"><a href="vocablist.asp">Vocabulary list</a></li>
 <li><a href="<%= MM_Logout %>">Logout</a></li>
```

```
 </ul>
 </div>
 <div class="content">
 <h2>User Blacklist Address</h2>
 <p>
<%
 set email=server.CreateObject("ADODB.Connection")
        set rs=server.CreateObject("ADODB.RecordSet")
        set rs1=server.CreateObject("ADODB.RecordSet")
        set rslog=server.CreateObject("ADODB.RecordSet")
        dim sqlstr
        dim sqlbl
        dim sqlwl
        logsess = Session("MM_Username")
        response.write("<font color=red><b>"& logsess &"@permail.com")
        mbox=Request("mbox")
        response.write(""& mbox &"")
        nspm=Request("nspm")
        response.write(""& nspm &"")
 page=request("page")
 email.open "permail"
 rslog.open "Select * from login where login= '"& logsess &"'",email
 user = "" & rslog(0) &""
        rs.open "SELECT blacklist.blid, blacklist.bladd FROM blacklist INNER JOIN
(login INNER JOIN userbl ON login.uid = userbl.uid) ON blacklist.blid = userbl.blid
where login.uid= "& user &"",email
 response.write("<p align=left>")
 on error resume next
        rs.pagesize=15
        response.write("<p align=left><table align=center cellspacing=2 cellpadding=0
border=0 >")
        Response.Write("<font color=#000000><tr bgcolor=#99CC99><th
align=center>No<th align=center>Blacklist Address")
        if len(page)>0 then
                rs.absolutepage=page
        else
                rs.absolutepage=1
        end if
        bgflip=true
        for i=1 to rs.pagesize
        if not rs.EOF then
                if bgflip then
                        Response.Write("<tr bgcolor=#eeeeee>")
                else
                        Response.Write("<tr bgcolor=#eeeeff>")
                end if
```

```
                bgflip=not bgflip
 Response.Write("<td align=left><li><td align=left><font face=Times New Roman
size=4> " & rs(1) & " ")
                rs.MoveNext
        end if
        next
        response.write("</table>")
        rs.Close
        rs1.Close
        email.Close
        set rs=nothing
        set rs1=nothing
        set email=nothing
        %>
  </p>
 <p> </p>
 <!-- end.content --></div>
 <div class="footer">
 <p align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font>
©</p>
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
```

User Vocabulary list

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>User Vocabulary list</title>
<link href="spam.css" rel="stylesheet" type="text/css" />
<link href="twoColLiqLtHdrinbx.css" rel="stylesheet" type="text/css" /><!--[if lte IE
7]>
<style>
.content { margin-right: -1px; } /* this 1px negative margin can be placed on any of the
columns in this layout with the same corrective effect. */
ul.nav a { zoom: 1; } /* the zoom property gives IE the hasLayout trigger it needs to
correct extra whiltespace between the links */
</style>
<![endif]-->
<style type="text/css">
.container.sidebar1.nav li a {
        font-family: Georgia, Times New Roman, Times, serif;
}
```

```
</style>
</head>
<body>
<div class="container">
 <div class="header">
 <div align="center"><strong>Personalized Spam Detection Email </strong></div>
 <!-- end.header --></div>
 <div class="sidebar1">
 <ul class="nav">
 <li><a href="login.asp">login by other user</a></li>
 <li><a href="inbox.asp">Inbox</a></li>
 <li><a href="junk.asp">Junk Mail</a></li>
 <li><a href="whitelist.asp">Whitelist</a></li>
 <li class="promo"><a href="blacklist.asp">Blacklist</a></li>
 <li><a href="<%= MM_Logout %>">Logout</a></li>
 </ul>
 </div>
 <div class="content">
 <h2>User Vocabulary List</h2>
 <p>
<%
 set email=server.CreateObject("ADODB.Connection")
        set rs=server.CreateObject("ADODB.RecordSet")
        set rs1=server.CreateObject("ADODB.RecordSet")
        set rslog=server.CreateObject("ADODB.RecordSet")
        dim sqlstr
        dim sqlbl
        dim sqlwl
        logsess = Session("MM_Username")
        response.write("<font color=red><b>"& logsess &"@permail.com")
        mbox=Request("mbox")
        response.write(""& mbox &"")
        nspm=Request("nspm")
        response.write(""& nspm &"")
 page=request("page")
 email.open "permail"
 rslog.open "Select * from login where login= '"& logsess &"'",email
 user = "" & rslog(0) &""
        rs.open "SELECT bvocab.vocid, vocab.word FROM vocab INNER JOIN (login
INNER JOIN ((junk INNER JOIN bvocab ON junk.jid = bvocab.jid) INNER JOIN
userjn ON junk.jid = userjn.jid) ON login.uid = userjn.uid) ON vocab.vocid =
bvocab.vocid WHERE (((login.uid)= "& user &"))",email
 response.write("<p align=left>")
 on error resume next
        rs.pagesize=15
```

```
        response.write("<p align=left><table align=center cellspacing=2 cellpadding=0
border=0 >")
        Response.Write("<font color=#000000><tr bgcolor=#99CC99><th
align=center>No<th align=center>Spam Vocabulary")
        if len(page)>0 then
                rs.absolutepage=page
        else
                rs.absolutepage=1
        end if
        bgflip=true
        for i=1 to rs.pagesize
        if not rs.EOF then
                if bgflip then
                        Response.Write("<tr bgcolor=#eeeeee>")
                else
                        Response.Write("<tr bgcolor=#eeeeff>")
                end if
                bgflip=not bgflip
 Response.Write("<td align=left><li><td align=left><font face=Times New Roman
size=5> " & rs(1) & " ")
                rs.MoveNext
        end if
        next
        response.write("</il></ol></td>")
        response.write("</table>")
        rs.Close
        rs1.Close
        email.Close
        set rs=nothing
        set rs1=nothing
        set email=nothing
        %>
  </p>
 <p> </p>
 <!-- end.content --></div>
 <div class="footer">
 <p align="center"><font face="Arial Black, Gadget, sans-serif"><font face="Georgia,
Times New Roman, Times, serif">All right resevied Personalized 2013</font></font>
©</p>
 <!-- end.footer --></div>
 <!-- end.container --></div>
</body>
</html>
```

C.2 Personalized spam detection algorithm interface code

c.1.1 Functions of algorithm
MATLAB code functions are as the following:

```
function vocabList = getVocabList(n)
%GETVOCABLIST reads the fixed vocabulary list in vocab.txt and returns a
%cell array of the words
% vocabList = GETVOCABLIST() reads the fixed vocabulary list in vocab.txt
% and returns a cell array of the words in vocabList.
 %% Read the fixed vocabulary list
fid = fopen('vocab.txt');
 % Store all dictionary words in cell array vocab{}
if n>1899
 n = 1899;
end
% For ease of implementation, we use a struct to map the strings => integers
% In practice, you will want to use some form of hashmap
vocabList = cell(n, 1);
for i = 1:n
 % Word Index (can ignore since it will be = i)
 fscanf(fid, '%d', 1);
 % Actual Word
 vocabList{i} = fscanf(fid, '%s', 1);
end
fclose(fid);
 end
==================================================
function stem = porterStemmer(inString)
% Porter, 1980, An algorithm for suffix stripping, Program, Vol. 14,
% no. 3, pp 130-137
% Original code modeled after the C version provided at:
% http://www.tartarus.org/~martin/PorterStemmer/c.txt
% The main part of the stemming algorithm starts here. b is an array of
% characters, holding the word to be stemmed. The letters are in b[k0],
% b[k0+1] ending at b[k]. In fact k0 = 1 in this demo program (since
% matlab begins indexing by 1 instead of 0). k is readjusted downwards as
% the stemming progresses. Zero termination is not in fact used in the
% algorithm.
 % To call this function, use the string to be stemmed as the input
% argument. This function returns the stemmed word as a string.
 % Lower case string
inString = lower(inString);
global j;
b = inString;
k = length(b);
k0 = 1;
j = k;
```

```
% With this if statement, strings of length 1 or 2 don't go through the  stemming process.
Remove this conditional to match the published  algorithm.
stem = b;
if k > 2
 % Output displays per step are commented out.
 %disp(sprintf('Word to stem: %s', b));
 x = step1ab(b, k, k0);
 %disp(sprintf('Steps 1A and B yield: %s', x{1}));
 x = step1c(x{1}, x{2}, k0);
 %disp(sprintf('Step 1C yields: %s', x{1}));
 x = step2(x{1}, x{2}, k0);
 %disp(sprintf('Step 2 yields: %s', x{1}));
 x = step3(x{1}, x{2}, k0);
 %disp(sprintf('Step 3 yields: %s', x{1}));
 x = step4(x{1}, x{2}, k0);
 %disp(sprintf('Step 4 yields: %s', x{1}));
 x = step5(x{1}, x{2}, k0);
 %disp(sprintf('Step 5 yields: %s', x{1}));
 stem = x{1};
end
% cons(j) is TRUE <=> b[j] is a consonant.
function c = cons(i, b, k0)
c = true;
switch(b(i))
 case {'a', 'e', 'i', 'o', 'u'}
 c = false;
 case 'y'
 if i == k0
 c = true;
  else
  c = ~cons(i - 1, b, k0);
 end
end
% mseq() measures the number of consonant sequences between k0 and j. If
% c is a consonant sequence and v a vowel sequence, and <..> indicates
% arbitrary presence,
 % <c><v> gives 0
% <c>vc<v> gives 1
% <c>vcvc<v> gives 2
% <c>vcvcvc<v> gives 3
function n = measure(b, k0)
global j;
n = 0;
i = k0;
while true
 if i > j
```

```
 return
 end
 if ~cons(i, b, k0)
 break;
 end
 i = i + 1;
end
i = i + 1;
while true
 while true
 if i > j
 return
    end
 if cons(i, b, k0)
 break;
    end
 i = i + 1;
 end
 i = i + 1;
 n = n + 1;
 while true
 if i > j
 return
    end
 if ~cons(i, b, k0)
 break;
    end
 i = i + 1;
 end
 i = i + 1;
end
% vowelinstem() is TRUE <=> k0,...j contains a vowel
function vis = vowelinstem(b, k0)
global j;
for i = k0:j,
 if ~cons(i, b, k0)
 vis = true;
 return
 end
end
vis = false;
%doublec(i) is TRUE <=> i,(i-1) contain a double consonant.
function dc = doublec(i, b, k0)
if i < k0+1
 dc = false;
 return
```

```
end
if b(i) ~= b(i-1)
 dc = false;
 return
end
dc = cons(i, b, k0);
% cvc(j) is TRUE <=> j-2,j-1,j has the form consonant - vowel - consonant
% and also if the second c is not w,x or y. this is used when trying to
% restore an e at the end of a short word. e.g.
%
% cav(e), lov(e), hop(e), crim(e), but
% snow, box, tray.
function c1 = cvc(i, b, k0)
if ((i < (k0+2)) || ~cons(i, b, k0) || cons(i-1, b, k0) || ~cons(i-2, b, k0))
 c1 = false;
else
 if (b(i) == 'w' || b(i) == 'x' || b(i) == 'y')
 c1 = false;
 return
 end
 c1 = true;
end
% ends(s) is TRUE <=> k0,...k ends with the string s.
function s = ends(str, b, k)
global j;
if (str(length(str)) ~= b(k))
 s = false;
 return
end % tiny speed-up
if (length(str) > k)
 s = false;
 return
end
if strcmp(b(k-length(str)+1:k), str)
 s = true;
 j = k - length(str);
 return
else
 s = false;
end
% setto(s) sets (j+1),...k to the characters in the string s, readjusting
% k accordingly.
function so = setto(s, b, k)
global j;
for i = j+1:(j+length(s))
 b(i) = s(i-j);
```

```
end
if k > j+length(s)
 b((j+length(s)+1):k) = '';
end
k = length(b);
so = {b, k};
% rs(s) is used further down.
% [Note: possible null/value for r if rs is called]
function r = rs(str, b, k, k0)
r = {b, k};
if measure(b, k0) > 0
 r = setto(str, b, k);
end
% step1ab() gets rid of plurals and -ed or -ing. e.g.
% caresses -> caress
% ponies ->  poni
% ties   ->  ti
% caress ->  caress
% cats   ->  cat
% feed   ->  feed
% agreed ->  agree
% disabled -> disable
% matting -> mat
% mating ->  mate
% meeting -> meet
% milling -> mill
% messing -> mess
% meetings -> meet
function s1ab = step1ab(b, k, k0)
global j;
if b(k) == 's'
 if ends('sses', b, k)
 k = k-2;
 elseif ends('ies', b, k)
 retVal = setto('i', b, k);
 b = retVal{1};
 k = retVal{2};
 elseif (b(k-1) ~= 's')
 k = k-1;
 end
end
if ends('eed', b, k)
 if measure(b, k0) > 0;
 k = k-1;
 end
elseif (ends('ed', b, k) || ends('ing', b, k)) && vowelinstem(b, k0)
```

```
 k = j;
 retVal = {b, k};
 if ends('at', b, k)
 retVal = setto('ate', b(k0:k), k);
 elseif ends('bl', b, k)
 retVal = setto('ble', b(k0:k), k);
 elseif ends('iz', b, k)
 retVal = setto('ize', b(k0:k), k);
 elseif doublec(k, b, k0)
 retVal = {b, k-1};
 if b(retVal{2}) == 'l' || b(retVal{2}) == 's' ||...
 b(retVal{2}) == 'z'
 retVal = {retVal{1}, retVal{2}+1};
 end
   elseif measure(b, k0) == 1 && cvc(k, b, k0)
 retVal = setto('e', b(k0:k), k);
 end
 k = retVal{2};
 b = retVal{1}(k0:k);
end
j = k;
s1ab = {b(k0:k), k};
% step1c() turns terminal y to i when there is another vowel in the stem.
function s1c = step1c(b, k, k0)
global j;
if ends('y', b, k) && vowelinstem(b, k0)
 b(k) = 'i';
end
j = k;
s1c = {b, k};
% step2() maps double suffices to single ones. so -ization ( = -ize plus
% -ation) maps to -ize etc. note that the string before the suffix must give
% m() > 0.
function s2 = step2(b, k, k0)
global j;
s2 = {b, k};
switch b(k-1)
 case {'a'}
 if ends('ational', b, k) s2 = rs('ate', b, k, k0);
 elseif ends('tional', b, k) s2 = rs('tion', b, k, k0); end;
 case {'c'}
 if ends('enci', b, k) s2 = rs('ence', b, k, k0);
 elseif ends('anci', b, k) s2 = rs('ance', b, k, k0); end;
 case {'e'}
 if ends('izer', b, k) s2 = rs('ize', b, k, k0); end;
 case {'l'}
```

```
 if ends('bli', b, k) s2 = rs('ble', b, k, k0);
 elseif ends('alli', b, k) s2 = rs('al', b, k, k0);
 elseif ends('entli', b, k) s2 = rs('ent', b, k, k0);
 elseif ends('eli', b, k) s2 = rs('e', b, k, k0);
 elseif ends('ousli', b, k) s2 = rs('ous', b, k, k0); end;
 case {'o'}
 if ends('ization', b, k) s2 = rs('ize', b, k, k0);
 elseif ends('ation', b, k) s2 = rs('ate', b, k, k0);
 elseif ends('ator', b, k) s2 = rs('ate', b, k, k0); end;
 case {'s'}
 if ends('alism', b, k) s2 = rs('al', b, k, k0);
 elseif ends('iveness', b, k) s2 = rs('ive', b, k, k0);
 elseif ends('fulness', b, k) s2 = rs('ful', b, k, k0);
 elseif ends('ousness', b, k) s2 = rs('ous', b, k, k0); end;
 case {'t'}
 if ends('aliti', b, k) s2 = rs('al', b, k, k0);
 elseif ends('iviti', b, k) s2 = rs('ive', b, k, k0);
 elseif ends('biliti', b, k) s2 = rs('ble', b, k, k0); end;
 case {'g'}
 if ends('logi', b, k) s2 = rs('log', b, k, k0); end;
end
j = s2{2};
% step3() deals with -ic-, -full, -ness etc. similar strategy to step2.
function s3 = step3(b, k, k0)
global j;
s3 = {b, k};
switch b(k)
 case {'e'}
 if ends('icate', b, k) s3 = rs('ic', b, k, k0);
 elseif ends('ative', b, k) s3 = rs('', b, k, k0);
 elseif ends('alize', b, k) s3 = rs('al', b, k, k0); end;
 case {'i'}
 if ends('iciti', b, k) s3 = rs('ic', b, k, k0); end;
 case {'l'}
 if ends('ical', b, k) s3 = rs('ic', b, k, k0);
 elseif ends('ful', b, k) s3 = rs('', b, k, k0); end;
 case {'s'}
 if ends('ness', b, k) s3 = rs('', b, k, k0); end;
end
j = s3{2};
% step4() takes off -ant, -ence etc., in context <c>vcvc<v>.
function s4 = step4(b, k, k0)
global j;
switch b(k-1)
 case {'a'}
 if ends('al', b, k) end;
```

```
 case {'c'}
 if ends('ance', b, k)
 elseif ends('ence', b, k) end;
 case {'e'}
 if ends('er', b, k) end;
 case {'i'}
 if ends('ic', b, k) end;
 case {'l'}
 if ends('able', b, k)
 elseif ends('ible', b, k) end;
 case {'n'}
 if ends('ant', b, k)
 elseif ends('ement', b, k)
 elseif ends('ment', b, k)
 elseif ends('ent', b, k) end;
 case {'o'}
 if ends('ion', b, k)
 if j == 0
  elseif ~(strcmp(b(j),'s') || strcmp(b(j),'t'))
 j = k;
        end
 elseif ends('ou', b, k) end;
 case {'s'}
 if ends('ism', b, k) end;
 case {'t'}
 if ends('ate', b, k)
 elseif ends('iti', b, k) end;
 case {'u'}
 if ends('ous', b, k) end;
 case {'v'}
 if ends('ive', b, k) end;
 case {'z'}
 if ends('ize', b, k) end;
end
if measure(b, k0) > 1
 s4 = {b(k0:j), j};
else
 s4 = {b(k0:k), k};
end
% step5() removes a final -e if m() > 1, and changes -ll to -l if m() > 1.
function s5 = step5(b, k, k0)
global j;
j = k;
if b(k) == 'e'
 a = measure(b, k0);
 if (a > 1) || ((a == 1) && ~cvc(k-1, b, k0))
```

```
 k = k-1;
 end
end
if (b(k) == 'l') && doublec(k, b, k0) && (measure(b, k0) > 1)
 k = k-1;
end
s5 = {b(k0:k), k};
=====================================================
function [eta I]= indicateEmail(email_contents,vocabList,vocabScore)
%PROCESSEMAIL preprocesses a the body of an email and
%returns a list of word_indices
% word_indices = PROCESSEMAIL(email_contents) preprocesses
% the body of an email and returns a list of indices of the
% words contained in the email.
%% S = 1;
% H = 1;
eta = 0;
% ============= Preprocess Email ==============
% Find the Headers ( \n\n and remove )
% Uncomment the following lines if you are working with raw emails with the
% full headers
% hdrstart = strfind(email_contents, ([char(10) char(10)]));
% email_contents = email_contents(hdrstart(1):end);
% Lower case
email_contents = lower(email_contents);
% Strip all HTML
% Looks for any expression that starts with < and ends with > and replace
% and does not have any < or > in the tag it with a space
email_contents = regexprep(email_contents, '<[^<>]+>', ' ');
% Handle Numbers
% Look for one or more characters between 0-9
email_contents = regexprep(email_contents, '[0-9]+', 'number');
% Handle URLS
% Look for strings starting with http:// or https://
email_contents = regexprep(email_contents,...
 '(http|https)://[^\s]*', 'httpaddr');
% Handle Email Addresses
% Look for strings with @ in the middle
email_contents = regexprep(email_contents, '[^\s]+@[^\s]+', 'emailaddr');
% Handle $ sign
email_contents = regexprep(email_contents, '[$]+', 'dollar');
% ============ Tokenize Email ===========
% Output the email to screen as well
%fprintf('\n==== Processed Email ====\n\n');
% Process file
 while ~isempty(email_contents)
```

```
% Tokenize and also get rid of any punctuation
[str, email_contents] =...
strtok(email_contents,...
[' @$/#.-:&*+=[]?!(){},""">_<;%' char(10) char(13)]);
% Remove any non-alphanumeric characters
str = regexprep(str, '[^a-zA-Z0-9]', '');
% Stem the word
% (the porterStemmer sometimes has issues, so we use a try catch block)
try str = porterStemmer(strtrim(str));
catch str = ''; continue;
end;
% Skip the word if it is too short
if length(str) < 1
continue;
end % Look up the word in the dictionary and add to word_indices if
% found
% Instructions: Fill in this function to add the index of str to
% word_indices if it is in the vocabulary. At this point
% of the code, you have a stemmed word from the email in
%   the variable str. You should look up str in the
%   vocabulary list (vocabList). If a match exists, you
%  should add the index of the word to the word_indices
vector. Concretely, if str = 'action', then you should
%   look up the vocabulary list to find where in vocabList
%   'action' appears. For example, if vocabList{18} =
%   'action', then, you should add 18 to the word_indices
%   vector (e.g., word_indices = [word_indices; 18]; ).
 % Note: vocabList{idx} returns a the word with index idx in the
%   vocabulary list.
 % Note: You can use strcmp(str1, str2) to compare two strings (str1 and
%  str2). It will return 1 only if the two strings are equivalent.
  %word_indices = [word_indices; find(strcmp(str,vocabList))];
ind = find(strcmp(str,vocabList));
if ~isempty(ind)
eta  = eta + log(1-vocabScore(ind))-log(vocabScore(ind));
% S = S * vocabScore(ind);
% H = H * (1-vocabScore(ind));
 end
end
% I = (1+S-H)/2;
 I = 1/(1+exp(eta));
% Print footer
% fprintf('\n\n=======================\n');
End
```

C.3 Arabic Spam Detection Program Code

```
import csv
import os
import math
import random
"""
utils.py
"""
#-------------------------------------------------------------------------
#no need stop words because ISRIStemmer add 60 Arabic stopwords
#stop_words = []
# Punctuation marks
forbidden_words = [",", "+", "&", "-", "_", ".", ")", "(", ":", "=", "/", """, "\"", "*"]
def counter(word_list):
 """Given a list of words, return a dictionary
 associating each word with the number of times it occurs"""
 counts = {}
 for word in word_list:
 # Intialise dictionary
 counts[word] = 0
 for word in word_list:
 # Calculate word counts
 counts[word] += 1
    return counts
def norm_dist(value, mean, sd):
 """Calculates the probability density of a normal distribution
 given the mean and standard deviation of the distribution."""
 if sd == 0.0:
 # If SD = 0, return a small non-zero number
 # as discussed in the report.
 return 0.05
 else:
 # PDF for normal distribution
 result = math.exp(-float((value - mean)**2) / (2.0*(sd**2))) * 1.0/(sd * math.sqrt(2 *
math.pi))
 return result
def mean(data):
 if data == []:
 return 0.0
 else:
 """Calculates the mean of a list"""
 sum = 0.0
 for item in data:
 sum += item
 #return str(float(sum)/len(data))
 return float(sum)/len(data)
def sd(data):
```

```
 if data == []:
 return 0.0
 else:
 "Calculates the standard deviation of a list"
 data_mean = mean(data)
 sums = 0.0
 for item in data:
 sums += (item-data_mean)**2
 return math.sqrt(float(sums)/(len(data)-1))
#--------------------------------------------------------------------------
#!/usr/bin/env python
# encoding: cp1256
"""
Message.py
"""
from utils import *
import math
import Stemmer
#--------------------------------------------------------------------------
class Message():
 """Implements the message class.
 Attributes
 subject - subject data
 body - body data
 subject word count - dictionary containing word --> count for subject
 body word count - dictionary containing word --> count for body
 spam - identifier if message is spam or non-spam"""
 def __init__(self, filename):
 # Initialise data
 file = open("./Data/" + filename, 'r')
 data = file.readlines()
 file.close()
 self.subject = data[0][9:].strip()
 self.body = [line.strip() for line in data[1:]][0]
      # Perform the Stemmer and numeric methods to further process the data
 self.stem_data()
 self.numeric_filter()
       # Calculate word counts for the data
 self.subject_word_count = counter(self.subject.split())
 self.body_word_count = counter(self.body.split())
      # Message attributes
 self.filename = filename
 self.spam = self.spam_class()
#-----------------------------------------------------------------------------
    def spam_class(self):
  """From the filename, classes the message as spam or non-spam"""
```

```
if self.filename[:6] == 'spamar':
return "Spam"
else:
 return "Non-spam"
def stem_data(self):
"""Stems the data, using ISRIStemmer algorithm"""
# The stemming object
stemmer = Stemmer.Stemmer()
def stem_string(string):
"""Input a string, returns a string with the
words replaced by their stemmed equivalents"""
        stemmed_list = []
for word in string.split():
stemmed_word = stemmer.stemWord(word)
stemmed_list.append(stemmed_word)
        stemmed_string = " ".join(stemmed_list)
return stemmed_string
      self.body = stem_string(self.body)
self.subject = stem_string(self.subject)
  def numeric_filter(self):
"""Replaces instances of numbers in a string with
a "NUMERIC" placeholder
e.g.("112", "22" ---> "NUMERIC")"""
def num_filter_string(string):
"""Input a string, returns a string with
strings of digits replaced with "NUMERIC"
"""
      filtered_list = []
for word in string.split():
if word.isdigit():
filtered_list.append("NUMERIC")
else:
        filtered_list.append(word)
         filtered_string = " ".join(filtered_list)
return filtered_string
      self.body = num_filter_string(self.body)
self.subject = num_filter_string(self.subject)
def tf_idf(self, corpus):
"""Input a corpus (with its list of document frequencies)
calculates the tf-idf score for the message for every feature"""
top100list = [(word, count) for count, word in corpus.top100]
   if corpus.type == "subject":
word_count = self.subject_word_count
else:
word_count = self.body_word_count
     self.tf_idf_scorelist = []
```

```
 # print word_count
 for word, document_frequency in top100list:
 if word not in word_count:
 # If word does not appear in the message, tf-idf == 0
 self.tf_idf_scorelist.append([word, 0])
 else:
     # calculate the tf-idf score for the word, appending the pair (word, score) to the list
 tf_idf_score = word_count[word] * math.log10(corpus.length /
float(document_frequency)) + 1.0/100
 self.tf_idf_scorelist.append([word, tf_idf_score])
      return self.tf_idf_scorelist
 #---------------------------------------------------------------------------
def testing():
 pass
#---------------------------------------------------------------------------
if __name__ == '__main__':
 testing()
-----------------------------------------------------------------------
# -*- coding: cp1256 -*-
#!/usr/local/bin/python
"""
Stemmer.py
"""
import unittest, re
from api import StemmerI
from isri import *
import codecs
#regexp = re.compile(r"[^aeiouy]*[aeiouy]+[^aeiouy](\w*)")
def isri_stem(word):
 isristem = ISRIStemmer()
 word = unicode(word,'cp1256')
 word = isristem.stem(word)
 return word
class Stemmer:
 """An instance of a stemming algorithm.
 When creating a Stemmer object, there is one required argument
 the appropriate stemming algorithm using ISRIStemming for Arabic
 language.
 """
max_cache_size = 10000
 def __init__ (self, cache_size=None):
 if cache_size:
 self.max_cache_size = cache_size
 def stemWord(self, word):
 """Stem a word.
```

The ISRI Stemmer requires that all tokens have Unicode string types. If you use Python IDLE on Arabic Windows you have to decode text first using Arabic '1256' coding.
"""
return Stemmer._stem(word)
def stemWords(self, words):
"""Stem a list of words.
This takes a single argument, words, which must be a sequence, iterator, generator or similar. The entries in words should either be UTF-8 encoded strings, or a unicode objects. The result is a list of the stemmed forms of the words. If the word supplied was a unicode object, the stemmed form will be a Unicode object: if the word supplied was a string, the stemmed form will be a UTF-8 encoded string.
"""
return [self.stemWord(word) for word in words]
@classmethod
def _stem(cls, word):
was_unicode = False
if isinstance(word, unicode):
was_unicode = True
word = word.encode('cp1256')
word = isri_stem(word)
if len(word) <= 2:
return word
#word = isri_stem(word)
if was_unicode:
return word.decode('cp1256')
return word
class TestISRIStem(unittest.TestCase):
def setUp(self):
pass
def testModule(self):
stemmer = Stemmer()
if __name__ == '__main__':
unittest.main()

```
# /usr/bin/env python
# encoding: cp1256
"""
Corpus.py
"""
from utils import *
from Message import *
import Stemmer
import nltk.tokenize.regexp
#----------------------------------------------------------------------------
class Corpus():
"""Corpus class. A superclass for the classes SubjectCorpus and BodyCorpus"""
```

```
def csv_write(self):
"""Writes a csv file
101 columns - 100 features and class identifier
191 rows - header (f1, f2...f100, class) and 190 examples"""
  headers = []
for index in xrange(len(self.top100)):
# Create the list [f1, f2,..., f100]
headers.append("f" + str(index + 1) )
     headers.append("Spam Class")
# Create the list [f1, f2..., f100, Spam Class]
     csv_file = []
csv_file.append(headers)
     for message in self.messages:
# Append the row of tf-idf scores for each feature
msg_scores = [scores[1] for scores in message.tf_idf_scorelist]
        # Append the spam class in the last column
msg_scores.append(message.spam)
        # Append the row to the file
csv_file.append(msg_scores)
     csv_filename = self.type + ".csv"
writer = csv.writer(open(csv_filename, "wb"))
for row in csv_file:
writer.writerow(row)
# Write the CSV file
   def get_length(self):
"""Find the number of examples in the corpus"""
self.length = len(self.data)
#-------------------------------------------------------------------------------------
def tf_idf_scores(self):
"""Calculate tf-idf scores for all messages in the corpus"""
for message in self.messages:
message.tf_idf(self)
   def DF_score(self):
"""Calculate the document frequency score for all words in the corpus"""
self.DF_counts = {}
     for message in self.cleaned_data:
for word in nltk.word_tokenize(message):
# Initialise the dictionary
self.DF_counts[word] = 0
     for message in self.cleaned_data:
word_added_already = []
for word in nltk.word_tokenize(message):
if word not in word_added_already:
# Avoids double counting a word if it appears twice in a message
self.DF_counts[word] += 1
word_added_already.append(word)
```

```
    word_list = sorted((value,key) for (key,value) in self.DF_counts.items())
# Sort our list, in order of least prevalent to most prevalent
    word_list.reverse()
# Reverse this list
    self.top100 = word_list[:100]
# Return the top 100 words
  return self.top100
def word_count(self):
"""Counts the number of unique words in the corpus"""
word_string = []
for message in self.data:
for words in nltk.word_tokenize(message):
word_string.append(words)
   word_counts = counter(word_string)
return word_counts
#---------------------------------------------------------------------------
def remove_stop_words(self):
"""Performs the filtering described in the data preprocessing
section of the report.
Removes punctuation
Stems words
Filters numeric data
"""
self.cleaned_data = []
stemmer = Stemmer.Stemmer()
for data in self.data:
words = data.split()
stemmed_words = [stemmer.stemWord(word) for word in words \
if word not in forbidden_words]
# Perhaps an overly complex line - returns a list of stemmed words,
# if the word is not a stop word or forbidden
words = []
for word in stemmed_words:
# Filters out our numeric features
#   e.g "112" --> "NUMERIC"
if word.isdigit():
words.append("NUMERIC")
else:
        words.append(word)
         clean_data = " ".join(words)
# Converts list to string
         self.cleaned_data.append(clean_data)
    return self.cleaned_data
#---------------------------------------------------------------------------
def creation(self):
"""A container method, performing the following operations:
```

```
 filtering  punctuation
 performing the stemming algorithm
 calculates tf-idf scores
 writes the CSV file
 """
 self.remove_stop_words()
 self.DF_score()
 self.tf_idf_scores()
 self.csv_write()
    print " - {0} CSV File For Arabic Email Created".format(self.type)
#--------------------------------------------------------------------------
class SubjectCorpus(Corpus):
 """Subject Corpus
 Message data is the subjects of the individual messages
 """
 def __init__(self, message_list):
 self.messages = message_list
 self.data = [message.subject for message in message_list]
 self.get_length()
 self.type = "Subject"
class BodyCorpus(Corpus):
 """Body Corpus
 Message data is the body of the individual messages
 """
 def __init__(self, message_list):
 self.messages = message_list
 self.data = [message.body for message in message_list]
 self.get_length()
 self.type = "Body"
#--------------------------------------------------------------------------
def Create_BC_SC_CSV():
 file_list = [(file, file[-3:]) for file in os.listdir("./Data")]
 proper_files = [file for file, extension in file_list if extension == "txt"]
 # Filters out files that are not text files
 message_list = [Message(file) for file in proper_files]
 # Our list of message objects
 SC = SubjectCorpus(message_list)
 SC.creation()
 BC = BodyCorpus(message_list)
 BC.creation()
if __name__ == '__main__':
 Create_B
C_SC_CSV()
=======================
# encoding: cp1256
#!/usr/bin/env python
```

```
"""
NaiveBayesClassifier.py
"""
import sys
import os
import csv
from utils import *
import math
import random
#-----------------------------------------------------------------------------
class NaiveBayesClassifier():
 """Naive Bayes Classifier class
 Implements the methods:
 CSV Read - reads a data file
 Train   - Trains on messages
 Feature_class_mean_sd - Calculates mean and sd
 for FEATURE when CLASS = SPAM CLASS
 Classify - Classifies a message
 P_spam_not_spam - Calculates probabilities a message
 is spam or non-spam
   Classification_test - tests if a message is correctly
 classified
       Stratification_test - Performs 10-fold cross validation"""
   def __init__(self, corpus):
 # Reads the corpus data
 self.type = corpus # Type of corpus - body or subject
 self.corpus_header, self.corpus_data = self.csv_read(corpus)
 self.corpus_data = self.cosine_normalisation()
#-----------------------------------------------------------------------------
 def csv_read(self, corpus):
 """Reads a CSV file. Outputs two lists:
 corpus_float_data - a list of messages
 corpus_header - a list of headers"""
 corpus_data = []
 corpus_file = self.type + ".csv" # e.g. subject.csv
 reader = csv.reader(open(corpus_file))
 for row in reader:
 # Scans through the rows, appending to the file
 corpus_data.append(row)
 corpus_header = corpus_data[:1] # Header data "f1, f2..."
 corpus_data = corpus_data[1:] # Message data with TF-IDF scores
 corpus_float_data = []
 for row in corpus_data:
 # Converts strings to floats
 float_row = [float(i) for i in row[:-1]]
 float_row.append(row[-1])
```

```
 corpus_float_data.append(float_row)
 return corpus_header, corpus_float_data
#-------------------------------------------------------------------------------------
 def cosine_normalisation(self):
 """Performs the cosine normalisation of data"""
   self.normalised_data = []
 for message in self.corpus_data:
 normalised_scores = []
 tf_idf_scores = message[:-1]
 normalisation_factor = math.sqrt(sum([i**2 for i in tf_idf_scores]))
 # Calculate \sum_{k} tf-idf(t_k, d_j)^2
 if normalisation_factor == 0:
 # Prevents dividing by zero
 self.normalised_data.append(message)
 else:
 for score in tf_idf_scores:
 normalised_scores.append(score/float(normalisation_factor))
 normalised_scores.append(message[-1])
 self.normalised_data.append(normalised_scores)
 return self.normalised_data
#-------------------------------------------------------------------------------------
   def train(self, training_set):
 """Trains the classifier by calculating the prior normal distribution
 parameters for the feature sets and TRUE/FALSE"""
 # The set of training messages
 training_messages = [self.corpus_data[i] for i in training_set]
      # Empty dictionary to hold mean and sd data
 self.mean_sd_data = {}
       for feature in range(100):
 self.mean_sd_data[feature] = {"Non-spam":[0, 0], "Spam":[0, 0]}
 for spam_class in ["Non-spam", "Spam"]:
 self.mean_sd_data[feature][spam_class] = []
 # Initialise the dictionary
       for feature in range(100):
 for spam_class in ["Non-spam", "Spam"]:
 # Fill the dictionary with values calculated from the feature_class_mean_sd method
 self.mean_sd_data[feature][spam_class] = self.feature_class_mean_sd(spam_class,
feature, training_messages)
       # Calculate the a-priori spam and non-spam probabilities
 spam_count = 0
 for message in training_messages:
 if message[-1] == "Spam":
 spam_count += 1
      self.mean_sd_data["Spam"] = spam_count / float(len(training_set))
 self.mean_sd_data["Non-spam"] = 1 - (spam_count / float(len(training_set)))
#-------------------------------------------------------------------------------------------------
```

```
   def feature_class_mean_sd(self, spam_class, feature, training_messages):
"""Calculates the mean and standard deviations for:
FEATURE when CLASS = SPAM CLASS"""
feature_list = []
for message in training_messages:
# Loop through all messages
if spam_class == message[-1]:
# If our message is in the right class
feature_list.append(message[feature])
# Take of the corresponding feature TF-IDF score
# Return the summary statistics of the relevant feature / class
return [mean(feature_list), sd(feature_list)]
#-------------------------------------------------------------------------------
   def classify(self, message):
"""Classify a message as spam or non-spam"""
# Probability that message is spam
p_spam = self.bayes_probability(message, "Spam")
# Probability that message is non-spam
p_not_spam = self.bayes_probability(message, "Non-spam")
      # print p_spam, p_not_spam
if p_spam > p_not_spam:
return "Spam"
# Message is spam
else:
 return "Non-spam"
# Message is non-spam
#-------------------------------------------------------------------------------
def bayes_probability(self, message, spam_class):
"""Probability that a message is or is not spam"""
      a_priori_class_probability = self.mean_sd_data[spam_class]
#Probability that a single message is spam or non-spam i.e. P(spam_id)
#print "Commencing Bayes Probability on Message 0"
#print "A priori Class Probability of {0} class is {1}".format(spam_class,
a_priori_class_probability)
class_bayes_probability = a_priori_class_probability
body_best_features = [ 6,8,11,34,35,45,48]
# Feature selection from WEKA
subject_best_features = range(1,100)
   if self.type == "Body":
"""Converts the features f1, f2,...fn into Python list indices"""
best_features = map(lambda x :x -1, body_best_features)
else:
 best_features = map(lambda x :x - 1, subject_best_features)
for feature in best_features:
# For all features
      message_tf_idf_score = message[feature]
```

```
 # Message tf_idf value
      tf_idf_mean = self.mean_sd_data[feature][spam_class][0]
 tf_idf_sd = self.mean_sd_data[feature][spam_class][1]
 # Get the parameters of the probability distribution governing this class
          prob_feature_given_class = norm_dist(message_tf_idf_score, tf_idf_mean,
tf_idf_sd)
 # Find the probabilty P(tf-idf_feature = score | msg_class = class)
 class_bayes_probability = class_bayes_probability * prob_feature_given_class
 # Multiply together to obtain total probabilitiy
 # as per the Naive Bayes independence assumption
return class_bayes_probability # Our probability that a message is spam or non-spam
    def classification_test(self, message):
 """Tests if a message is correctly classified"""
 if self.classify(message) == message[-1]:
 return True
 else:
  return False
   def stratification_test(self):
 """Performs 10-fold stratified cross validation"""
 already_tested = []
 test_set = []
 for i in range(10):
 """Create the set of 10 sixtylement random bins"""
 sample = random.sample([i for i in range(190) if i not in already_tested], 19)
 already_tested.extend(sample)
 test_set.append(sample)
       results = []
 for validation_data in test_set:
 """Create the training set (171 elements) and the validation data (19 elements)"""
 training_sets = [training_set for training_set in test_set if training_set is not
validation_data]
 training_data = []
 for training_set in training_sets:
 training_data.extend(training_set)
          self.train(training_data)
 # Train the probabilities of the Bayes Filter
          count = 0
 for index in validation_data:
 """Calculate the percentage of successful classifications"""
 if self.classification_test(self.corpus_data[index]):
 count += 1
        results.append(float(count)/len(validation_data))
 return results
#--------------------------------------------------------------------------
def print_results(results):
 """Formats results and prints them, along with summary statistic"""
```

```
 for result, index in zip(results, range(len(results))):
 print "Stratification Set {0} \t {1:.1f}% Classified Correctly.".format(index+1,
result*100)
 print "**"*30
 print "--"*30
 print "\n\tOverall Accuracy is {0:.1f}%".format(mean(results) * 100)
if __name__ == '__main__':
 import random
 random.seed(18)
 #  Sets the seed, for result reproducibility
 test = NaiveBayesClassifier("subject")
 print "\tTesting Arabic Subject Corpus"
 print "**"*30
 print "--"*30
 results = test.stratification_test()
 print_results(results)
 print
 print "\tTesting Body Arabic Corpus"
 print "**"*30
 print "--"*30
 test = NaiveBayesClassifier("body")
 results = test.stratification_test()
 print_results(results)
```

Appendix D
Screenshots

In this section we present screen shots of the implemented Permail systems

D.1 Permail General Screenshots

Figure D.1 Permail user home page. The home page it gives the user the ability to enter the mail system.

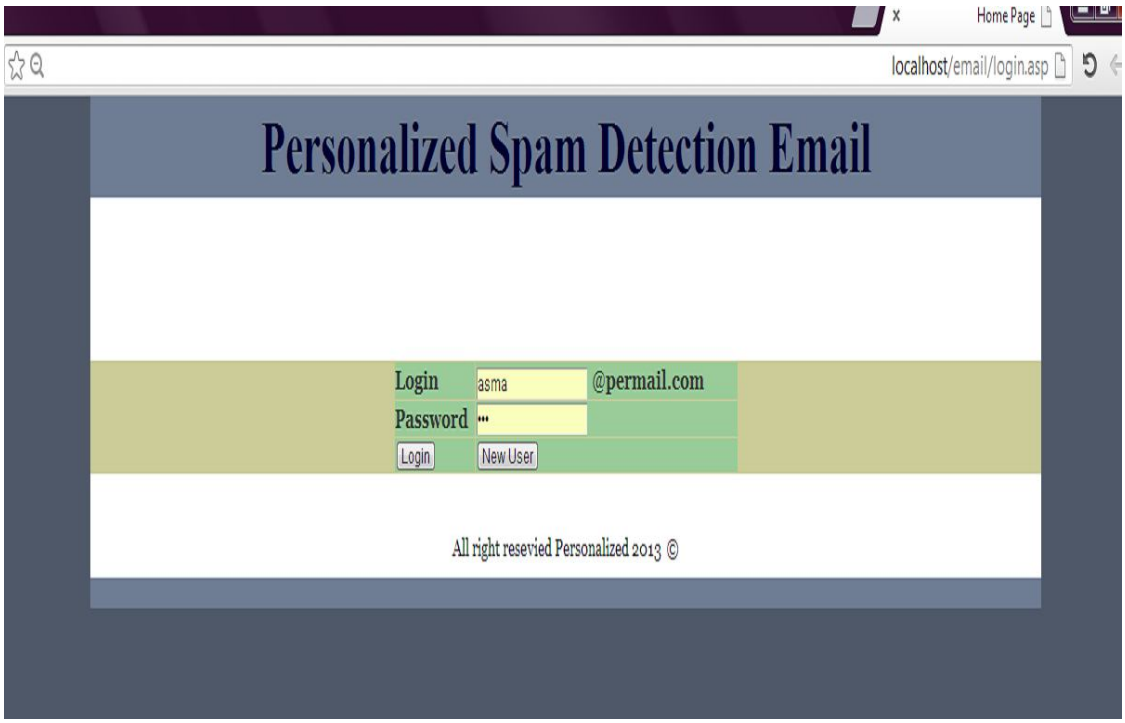


Figure D.2 Permail login user page

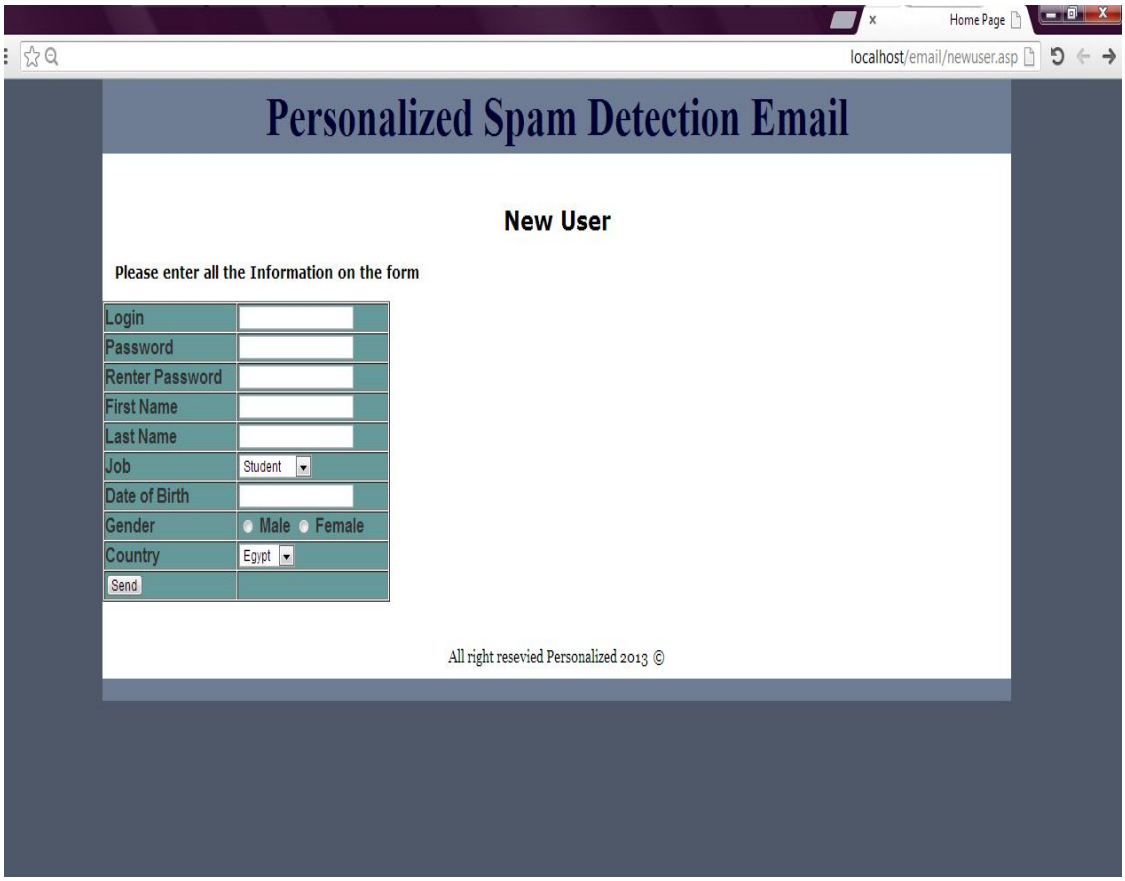


Figure D.3 Permail spam detection new user. User can create a new account by entering his full information on the form.

D.2 User Inbox Screenshots

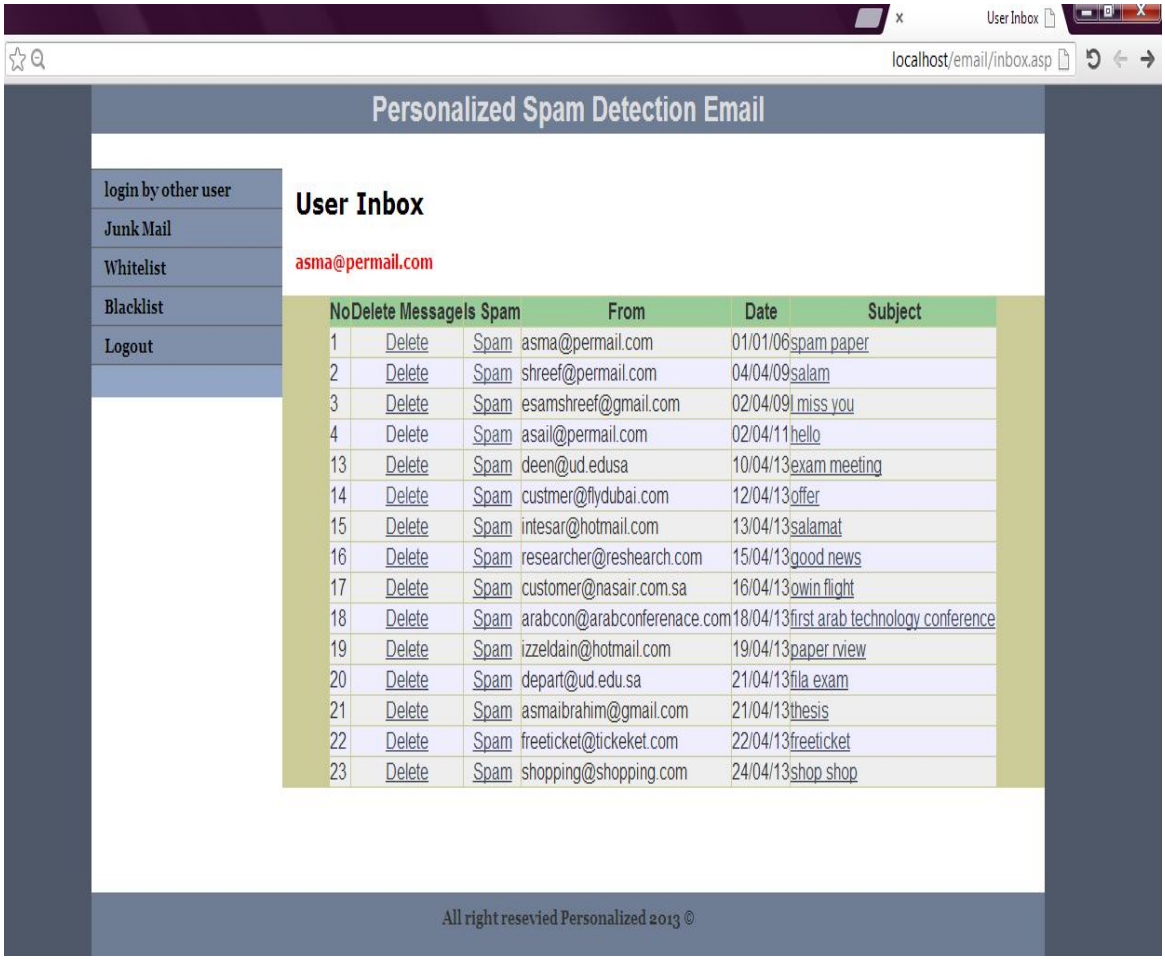


Figure D.4 Permail user Inbox (a). This page retrieves all messages from user Inbox.

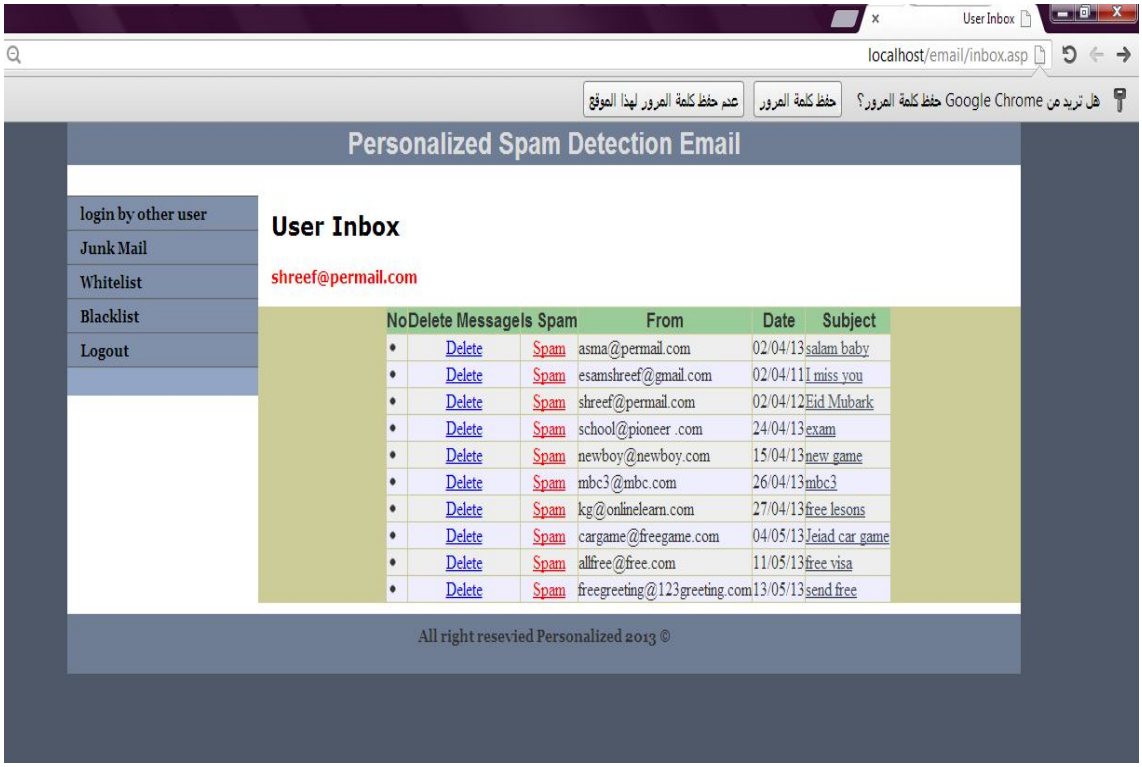


Figure D.4 Permail user Inbox (b). This page retrieves all messages from user Inbox.

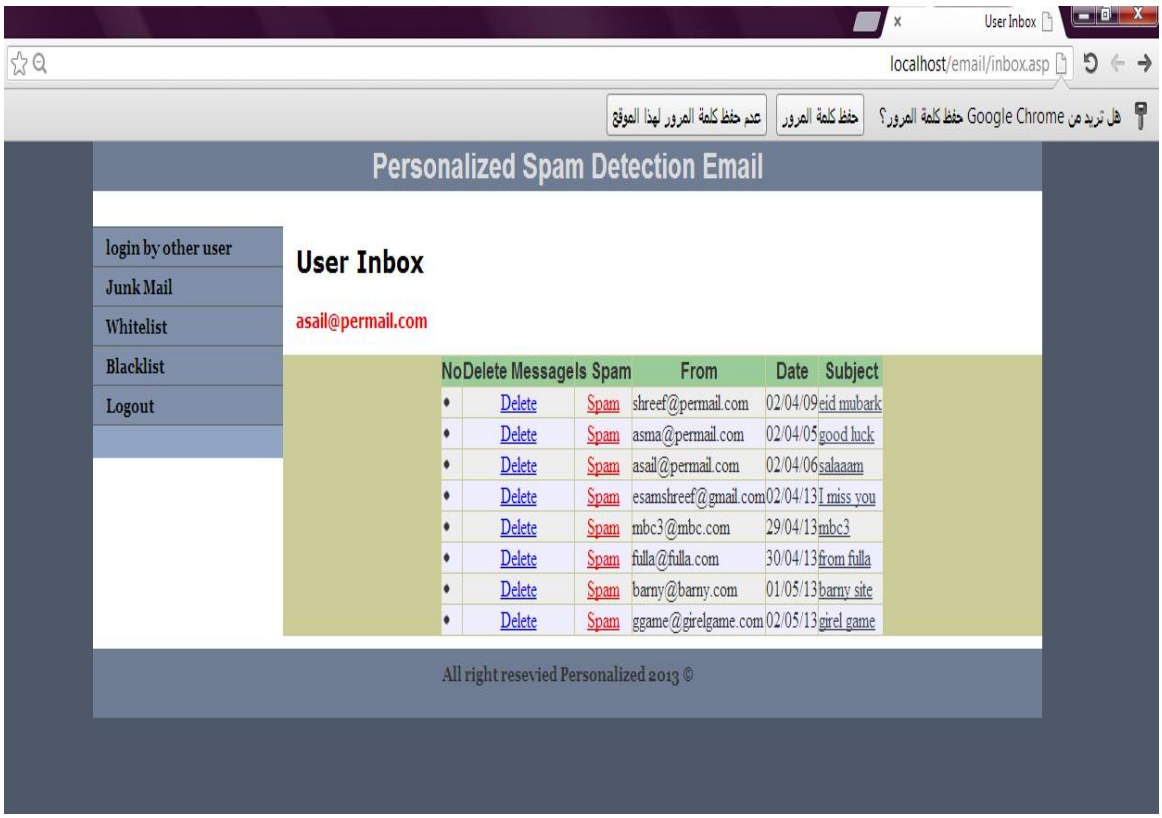


Figure D.4 Permail user Inbox (c). This page retrieves all messages from user Inbox.

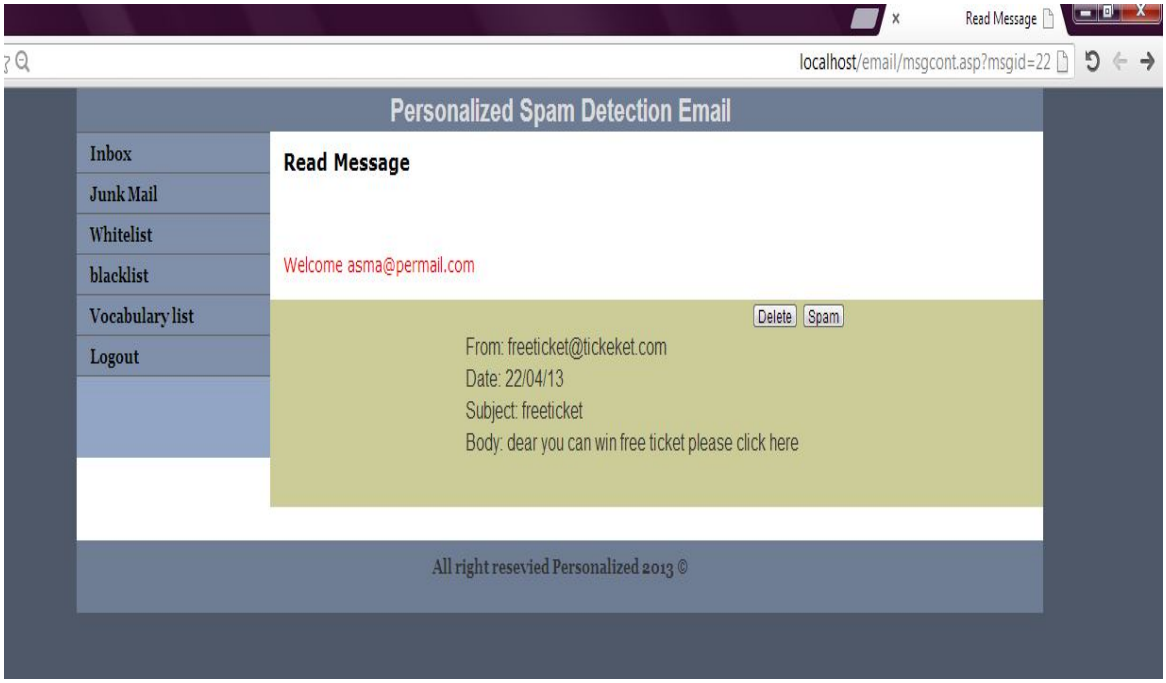


Figure D.5 User Read Message. When user click on the subject of any message he can read this message.

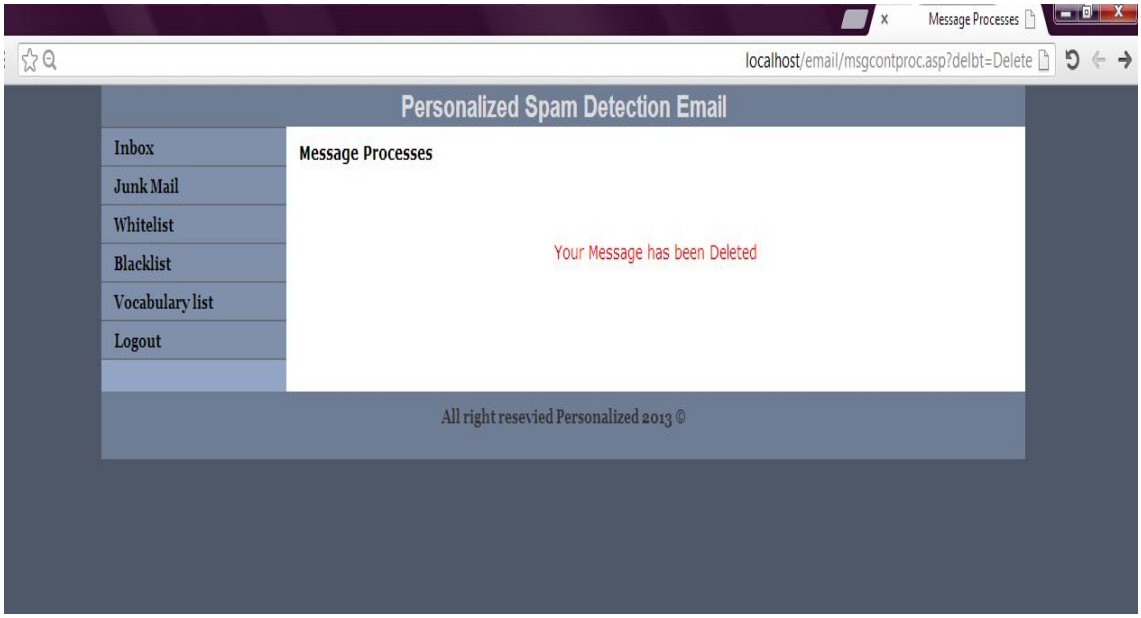


Figure D.6 User Delete Message

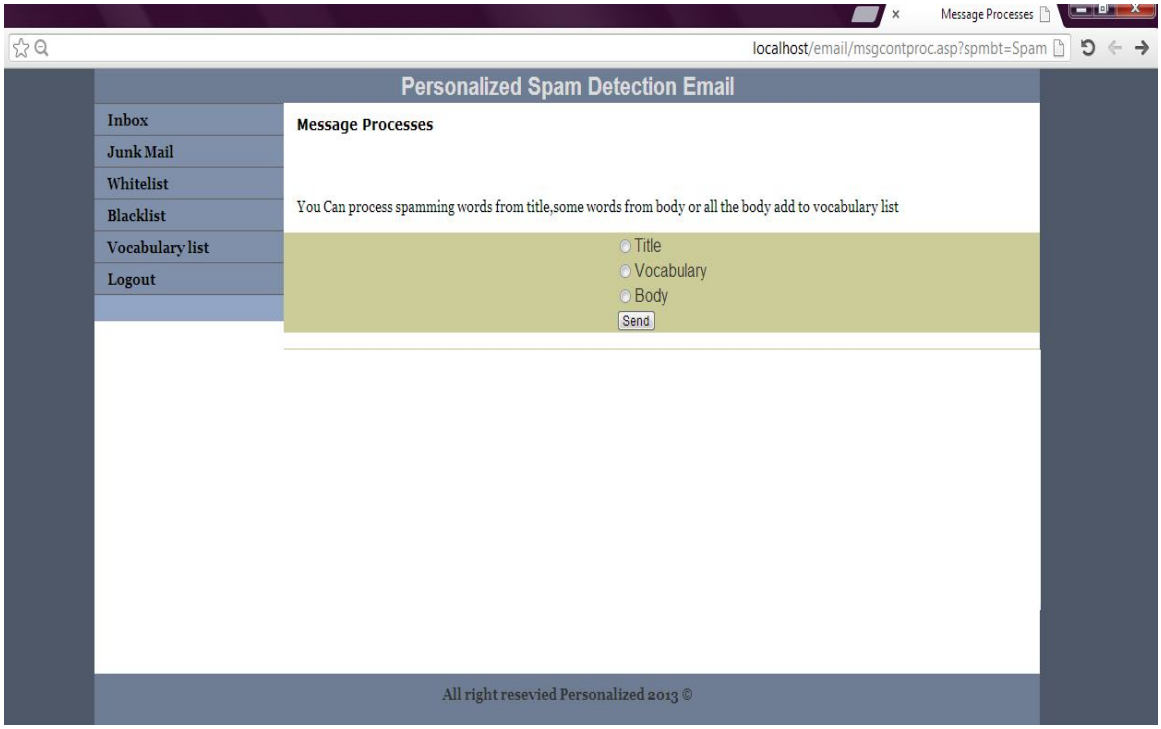


Figure D.7 Message process

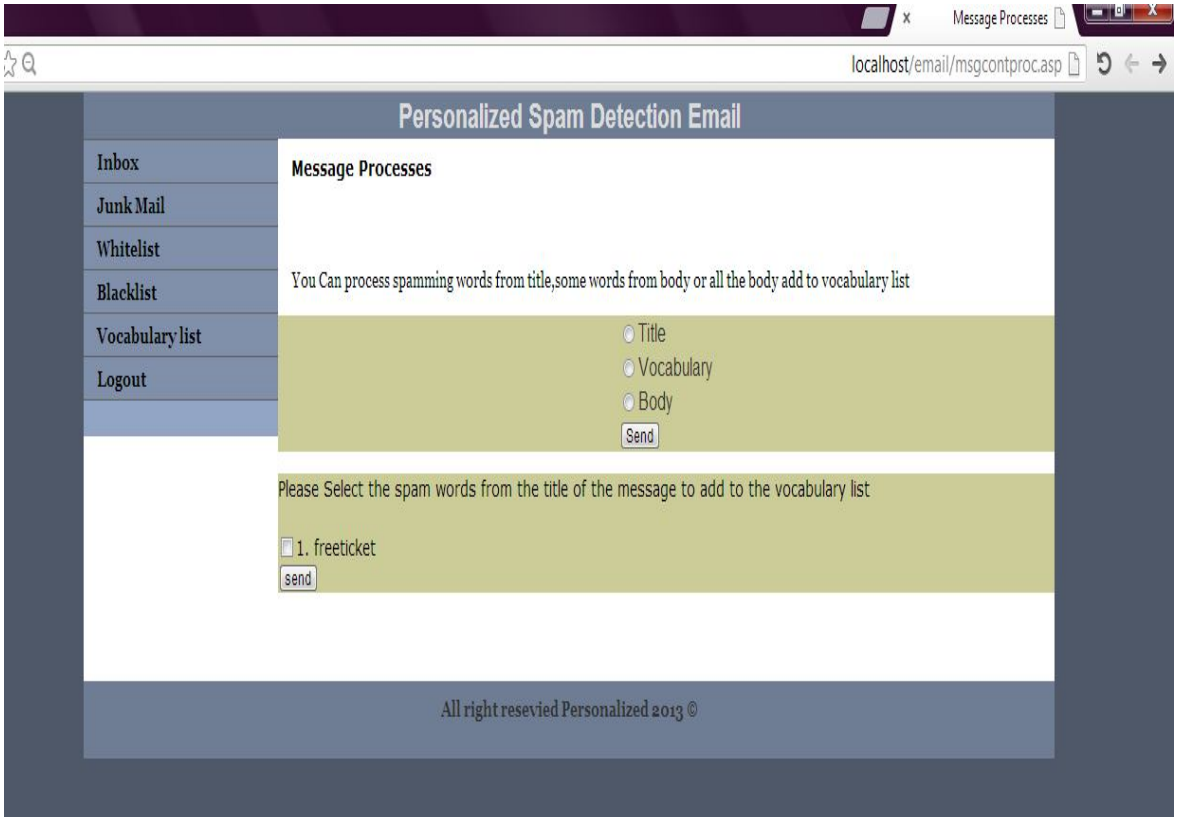


Figure D.8 The user selects title words to delete message

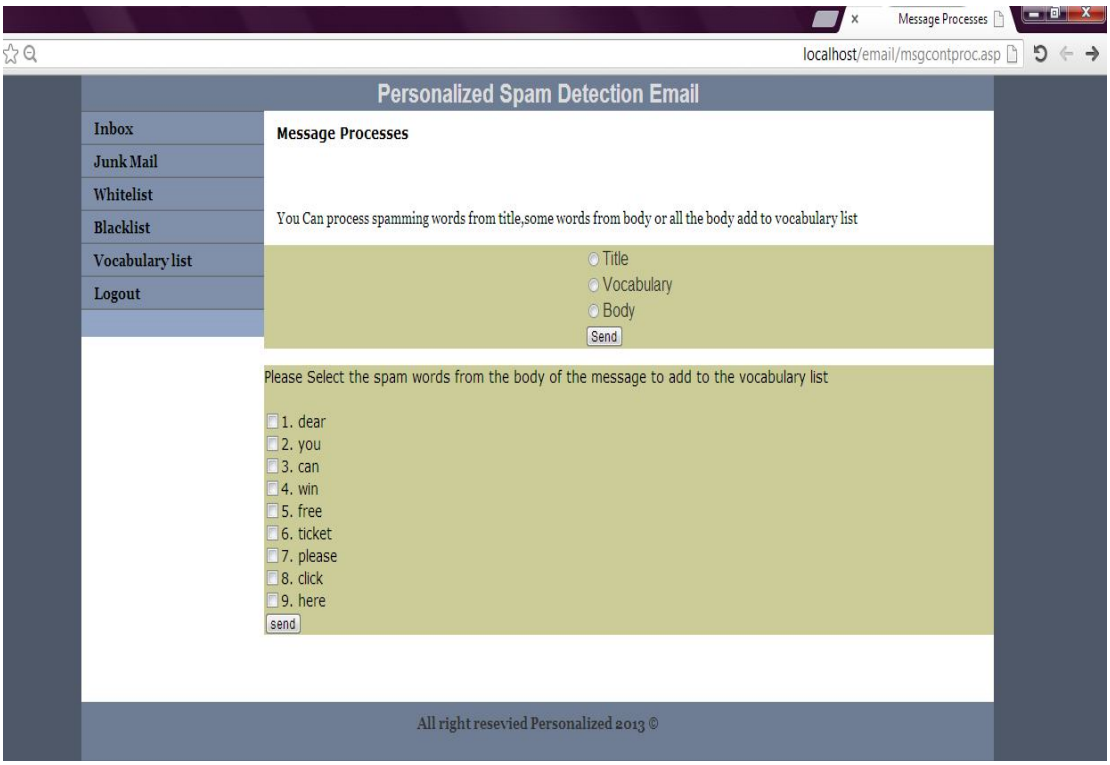


Figure D.9 The user selects body words to delete message

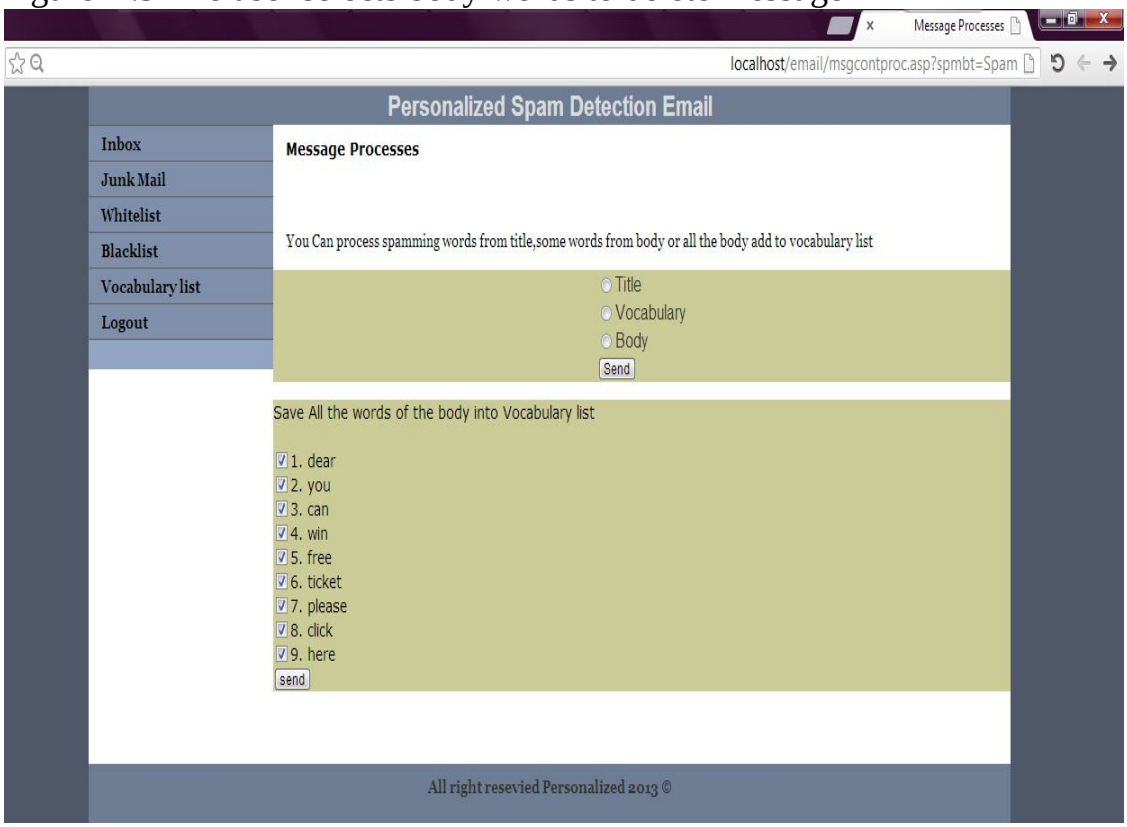


Figure D.10 The user selects all body words to delete message.

D.3 User junk mail Screenshots

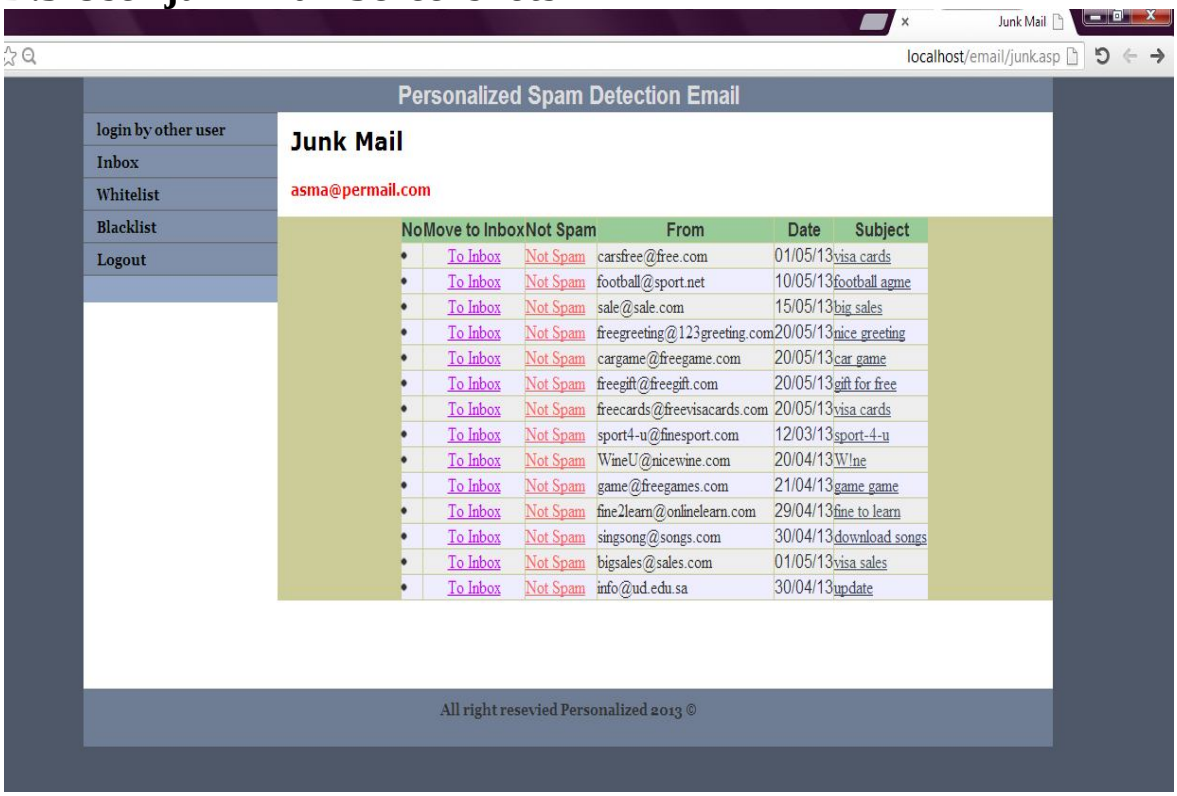


Figure D.11 Permail user's junk mail (a). This page retrieves all messages from the user's junk mail.

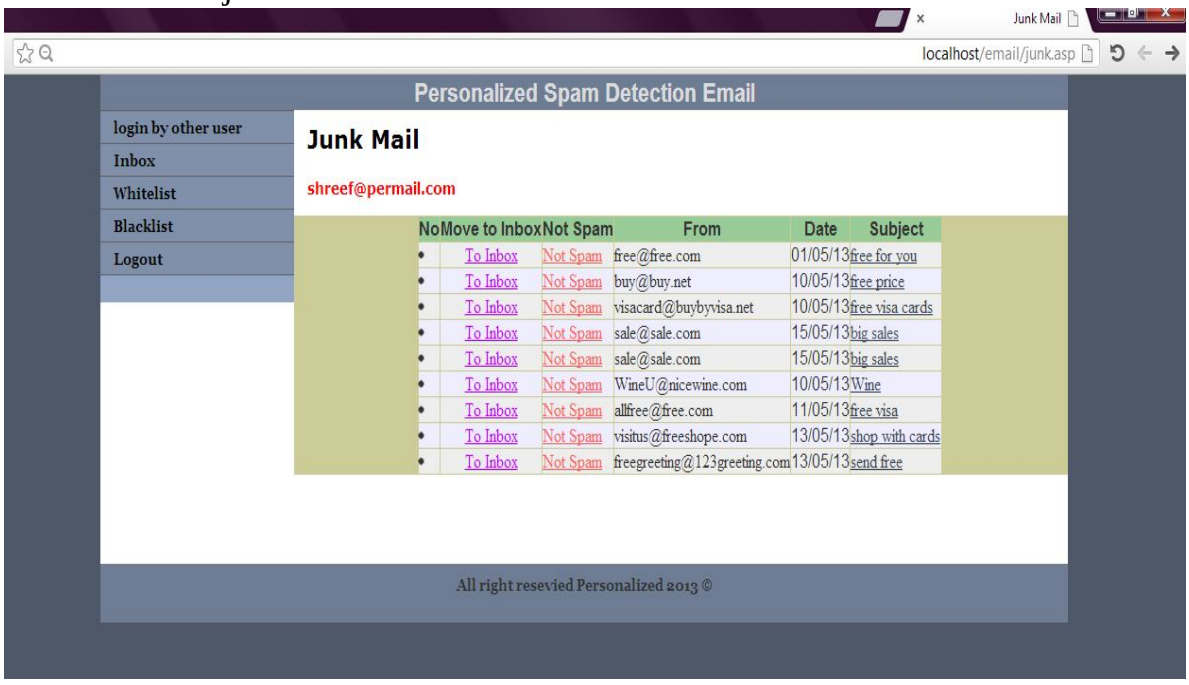


Figure D.11 Permail user's junk mail (b). This page retrieves all messages from the user's junk mail.

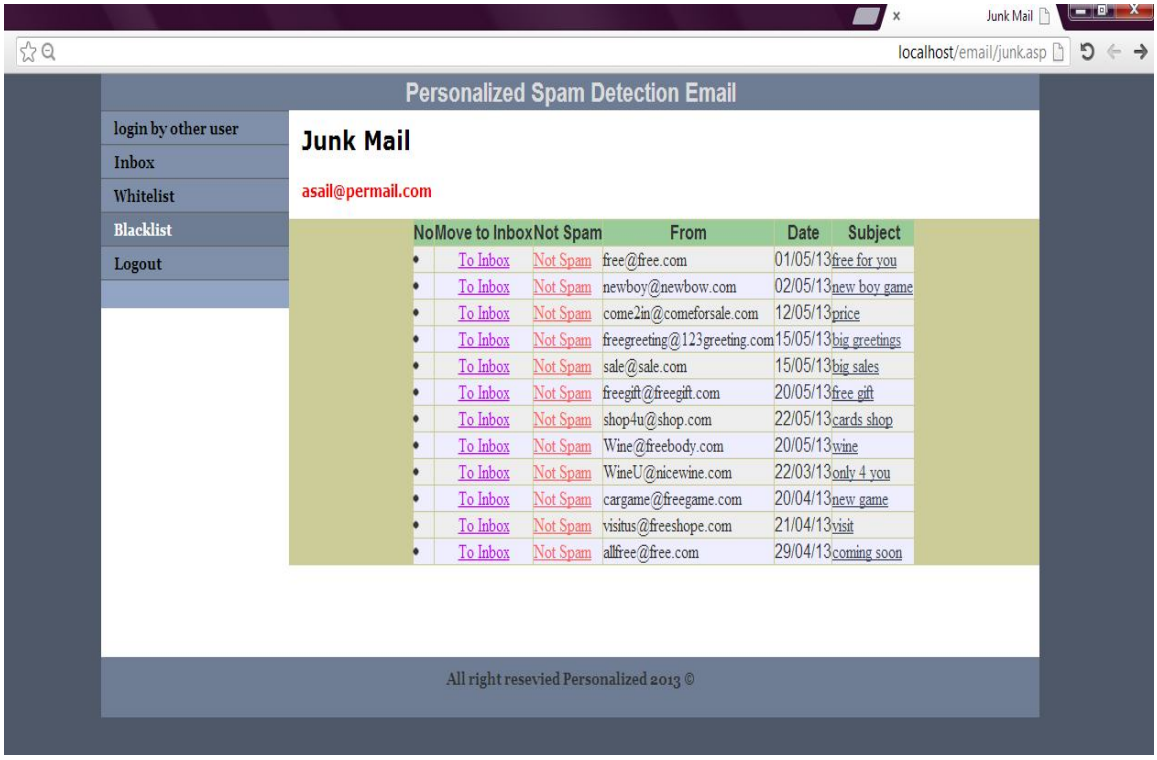


Figure D.11 Permail user's junk mail (c). This page retrieves all messages from the user's junk mail.

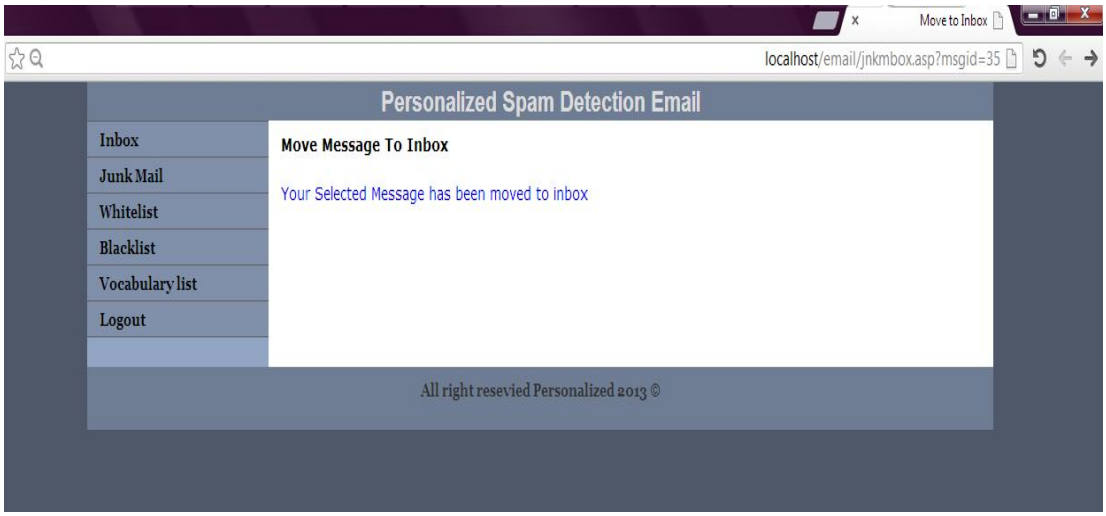


Figure D.12 Message has been moved to the Inbox

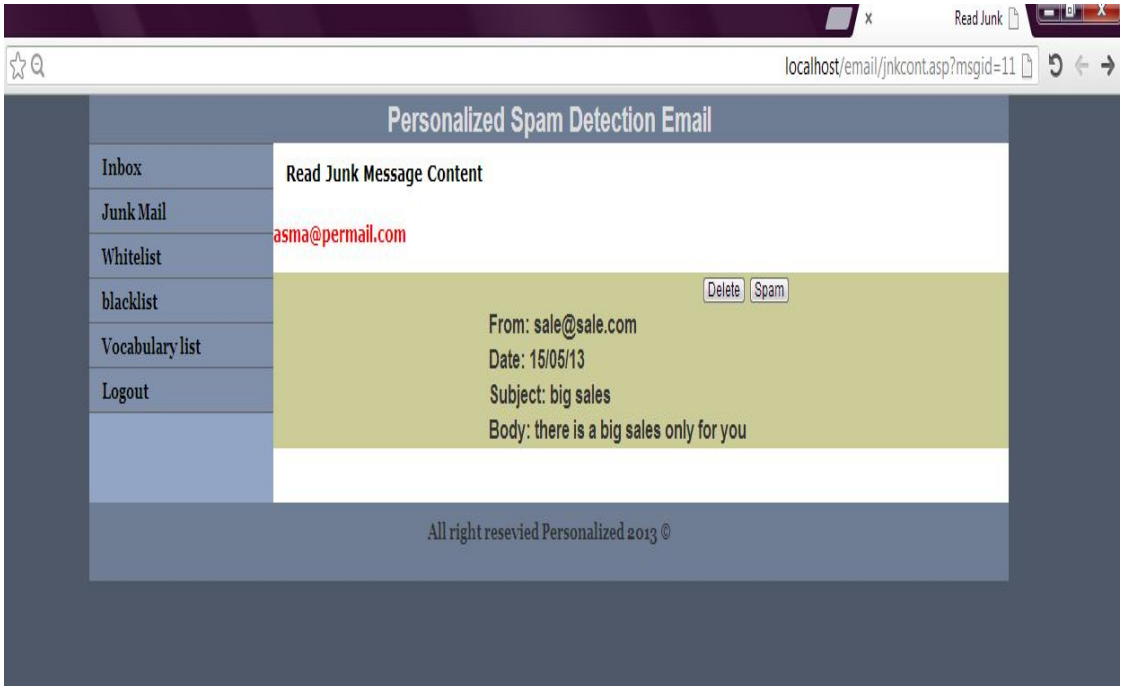


Figure D.13 User reads Junk Message

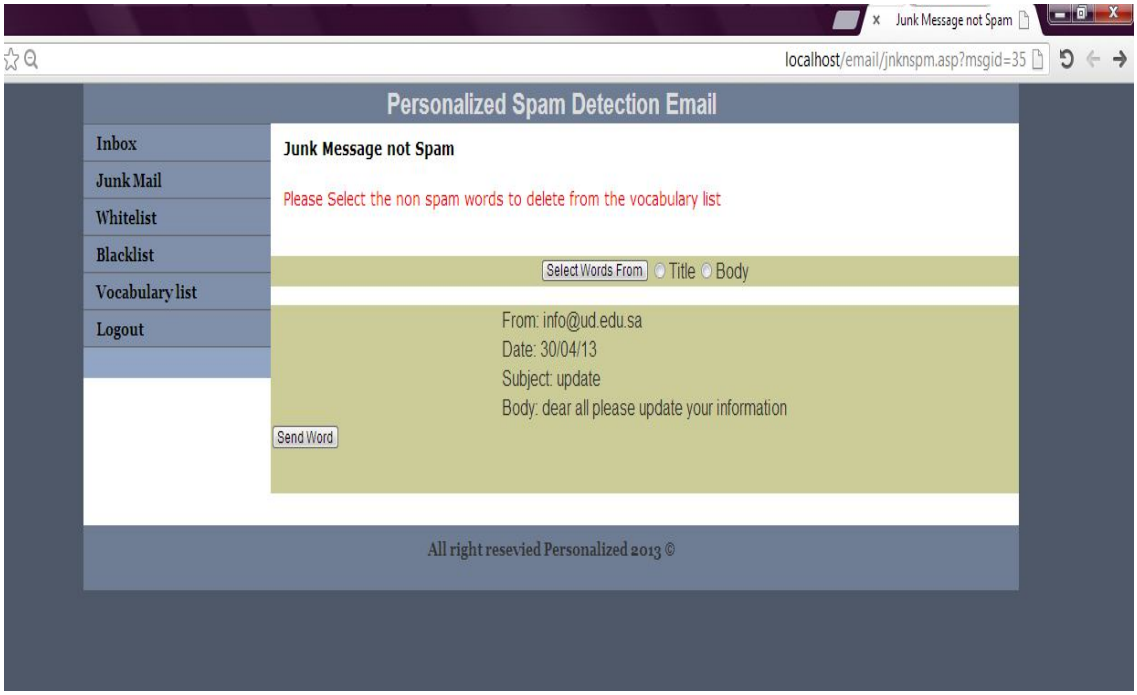


Figure D.14 Junk message not spam

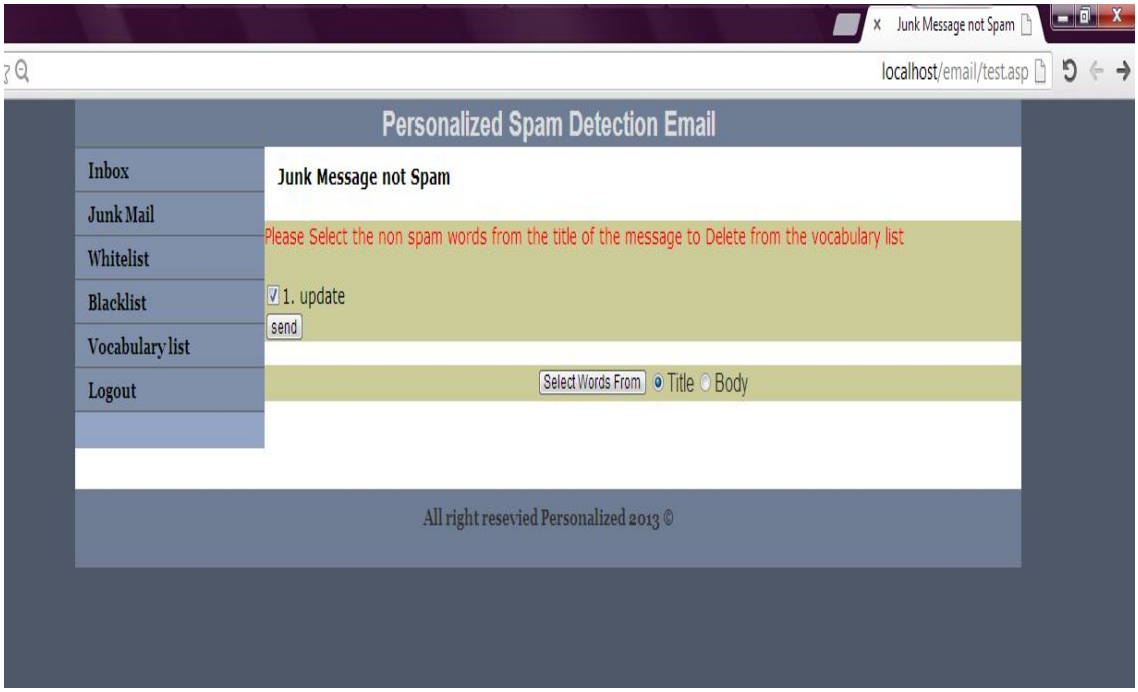


Figure D.15 Junk message not spam from the title

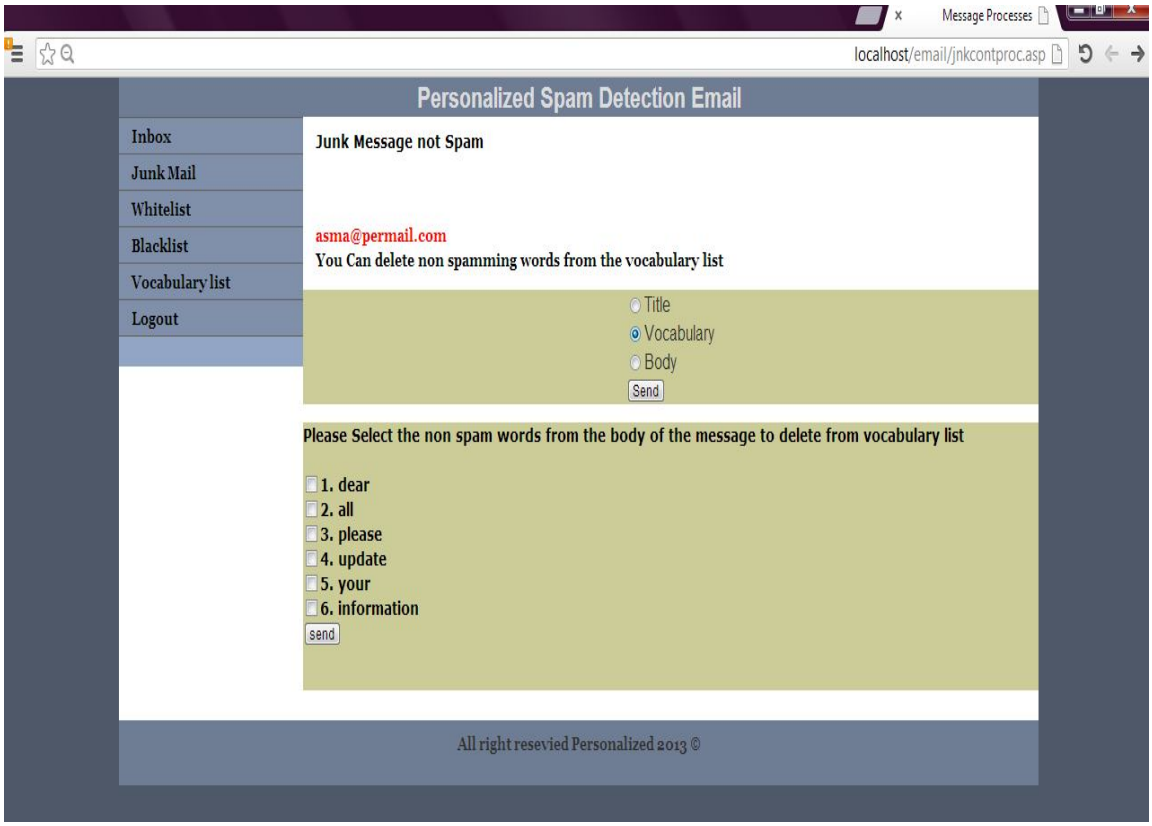


Figure D.16 Junk message not spam from vocabulary words

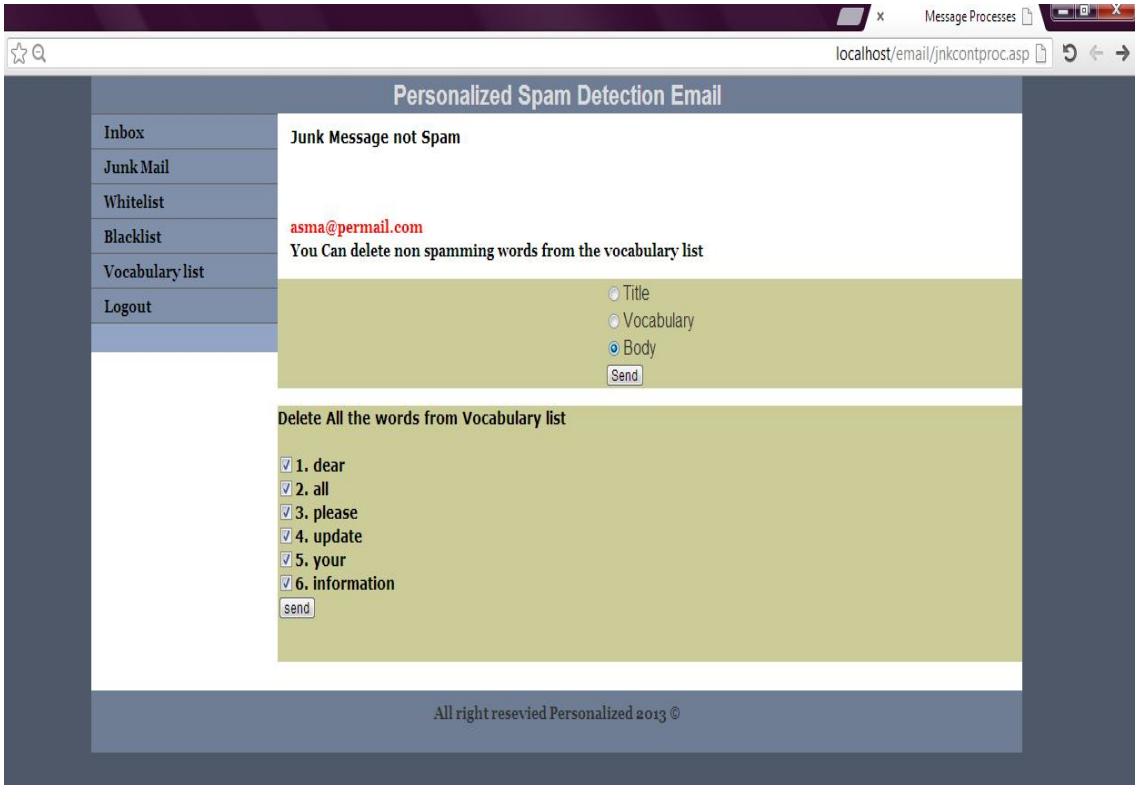


Figure D.17 Junk message not spam from all body

D.4 User lists Screenshots

This section presents user whitelist, blacklist and vocabulary lists

D.4.1 Whitelist

The following figures show the different whitelist between users a, b, and c.

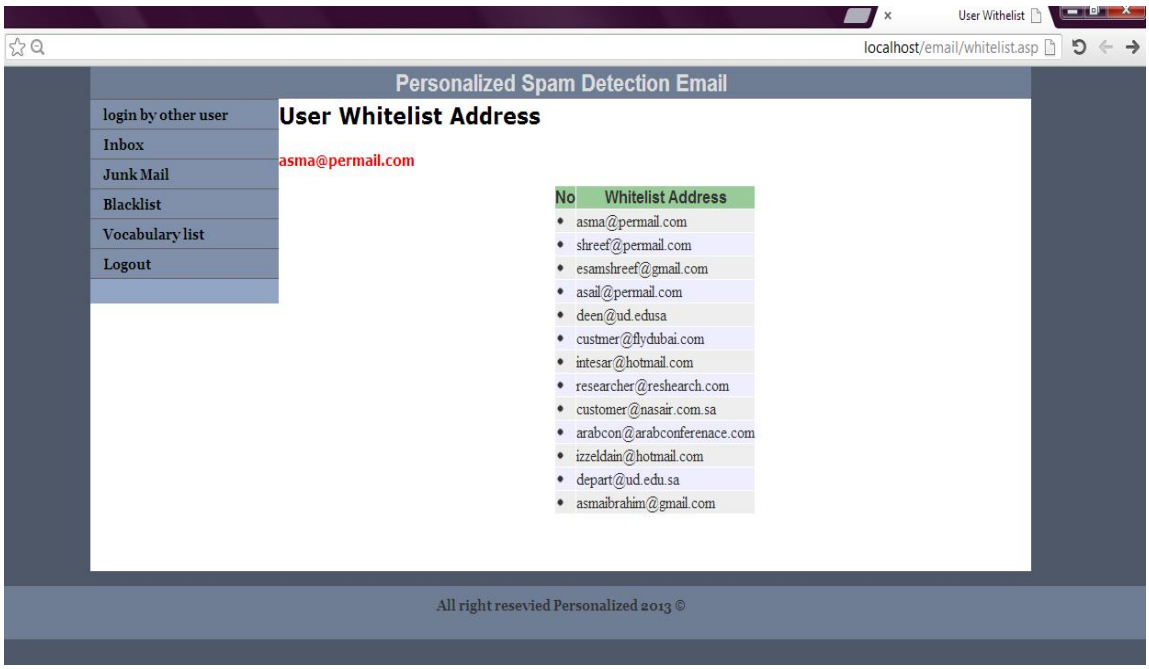


Figure D.18 User (a) whitelist examples

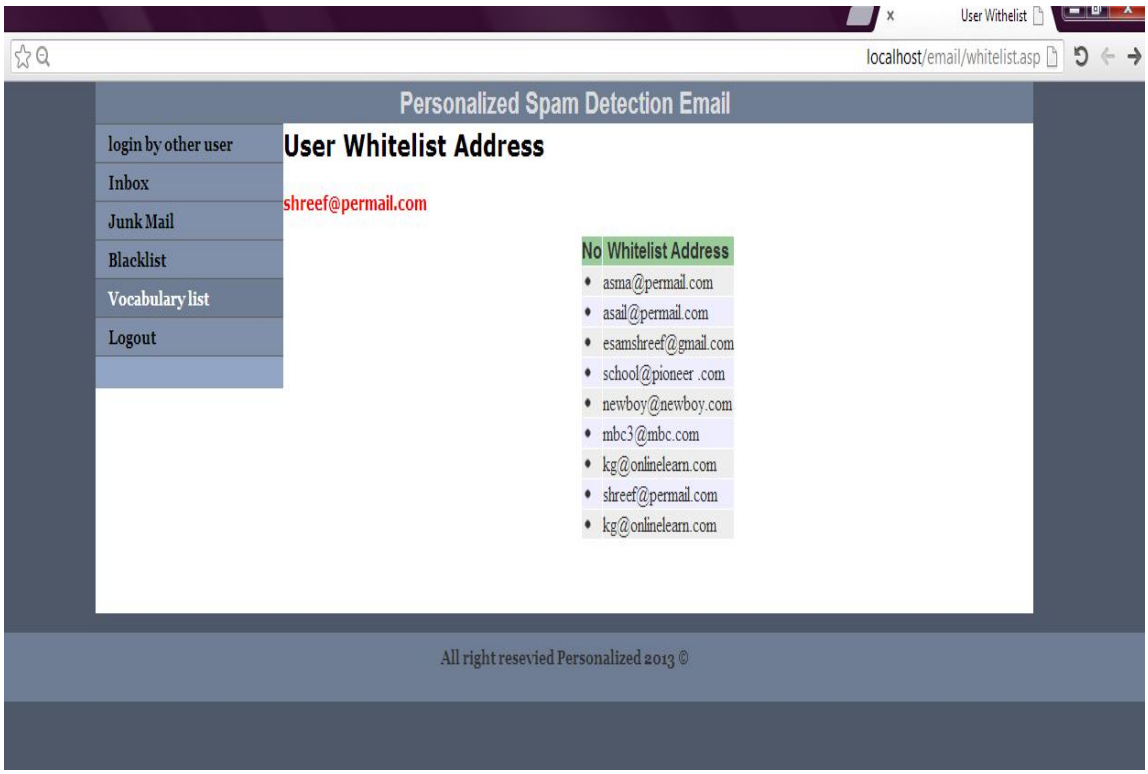


Figure D.18 User (b) whitelist examples

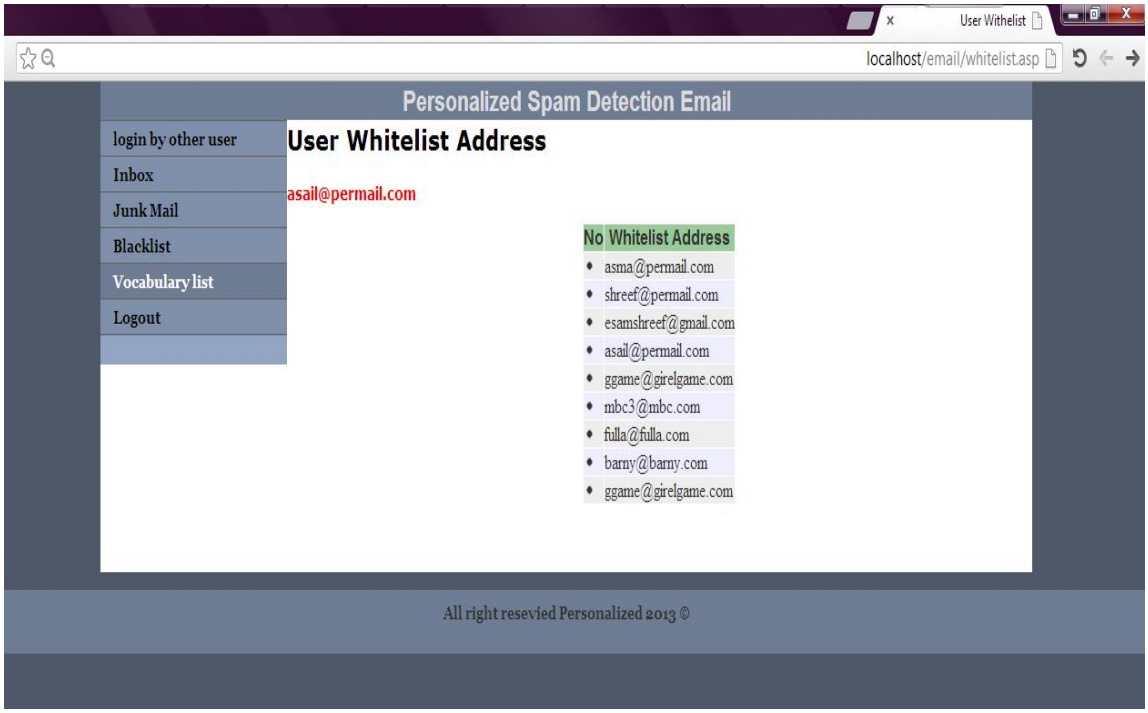


Figure D.18 User (c) whitelist examples

D.4.2 Blacklist

The following figures show the different blacklist between users a, b, and c.

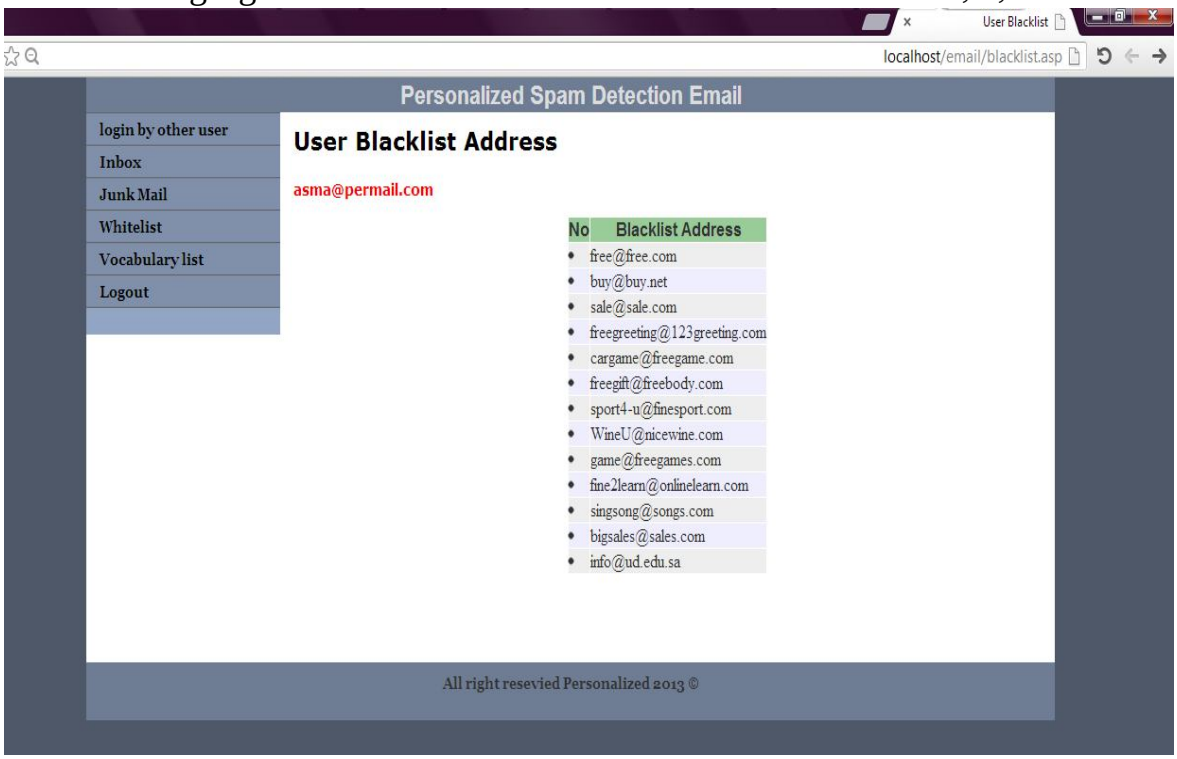


Figure D.19 User (a) blacklist example

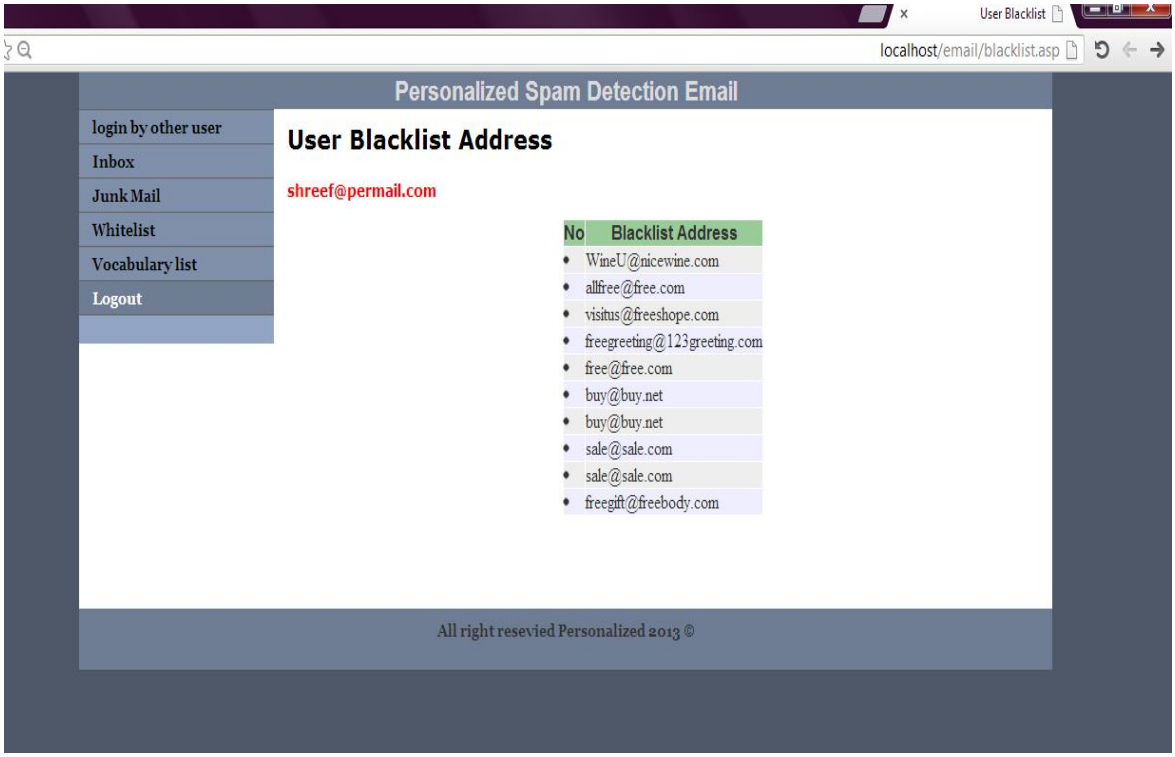


Figure D.19 User (b) blacklist example

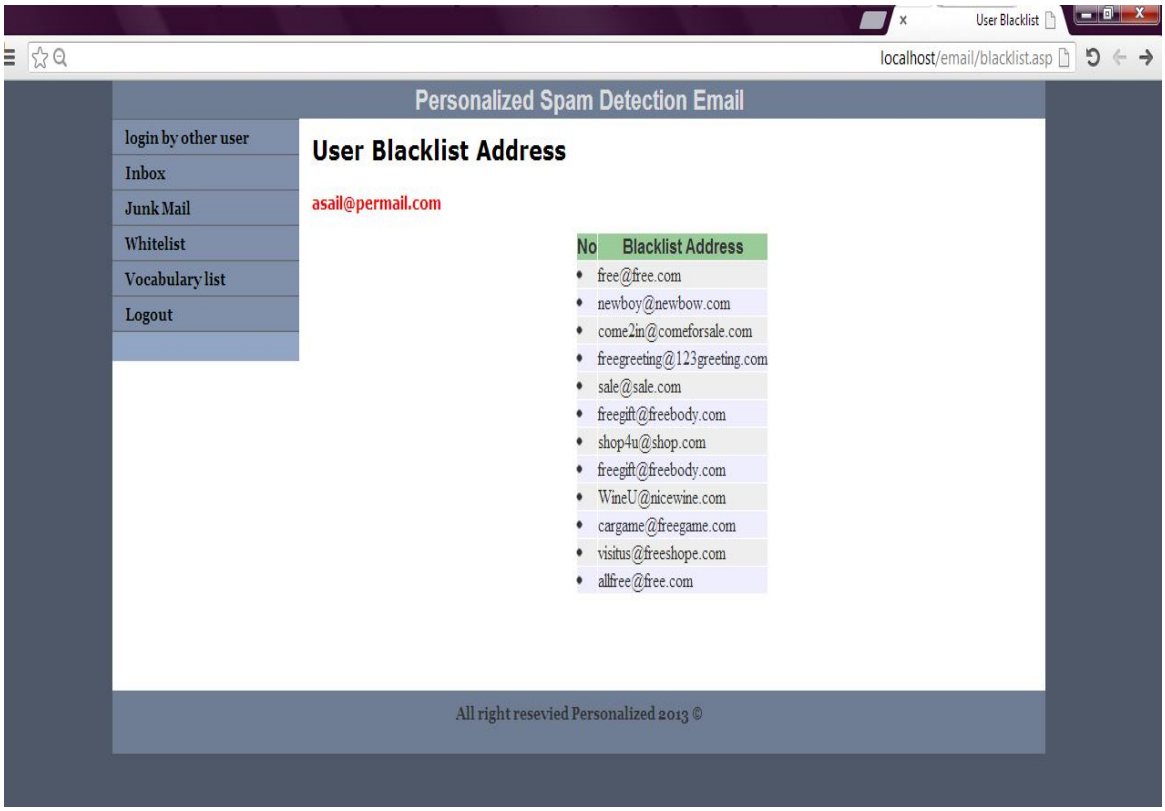


Figure D.19 User (c) blacklist examples

D.4.3 Vocabulary list

The following figures show the different vocabulary list between users

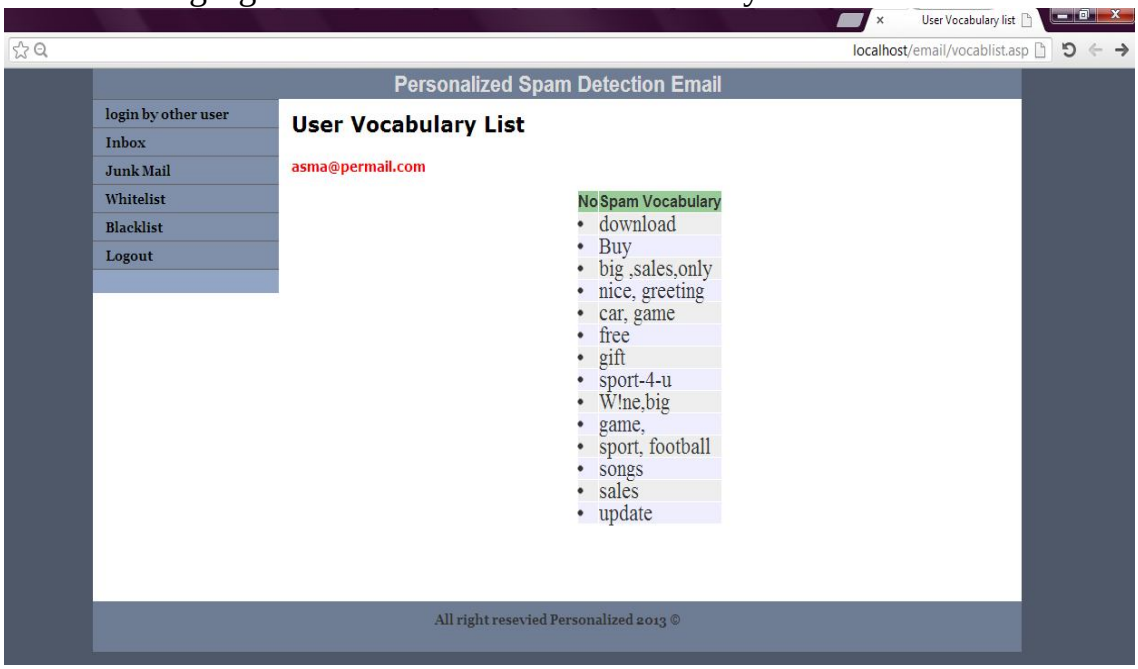


Figure D.20 User (a) vocabulary list example

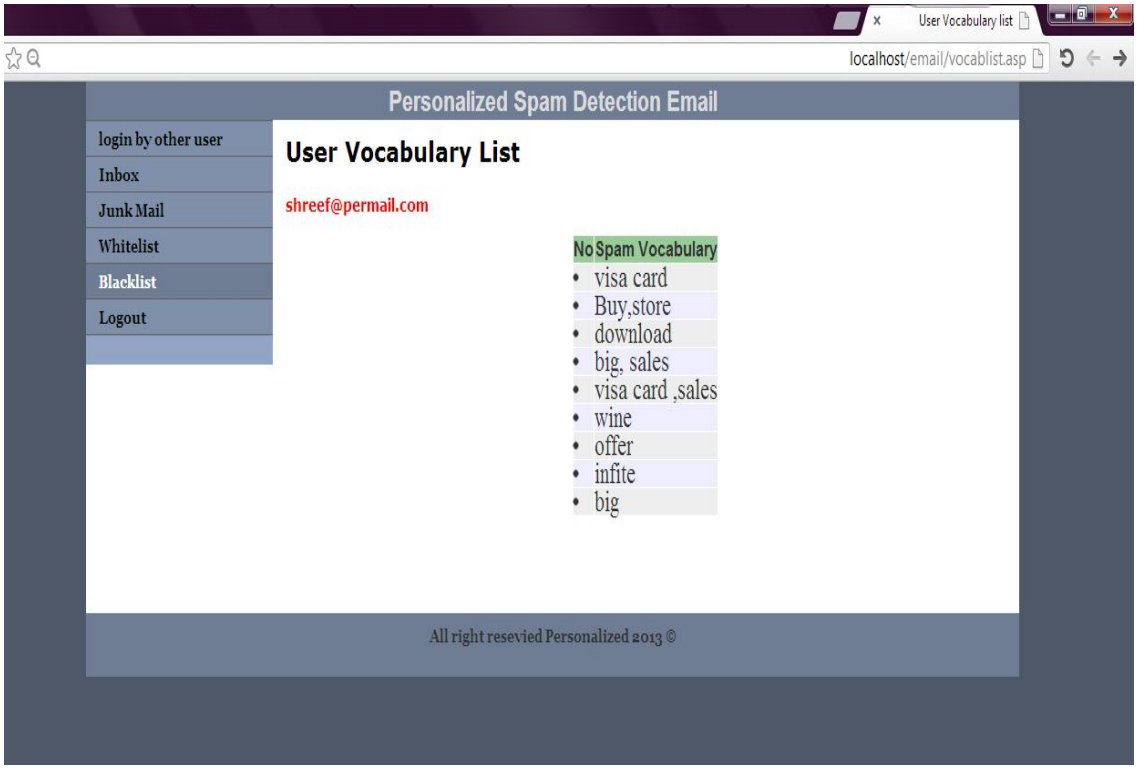


Figure D.20 User (b) vocabulary list example

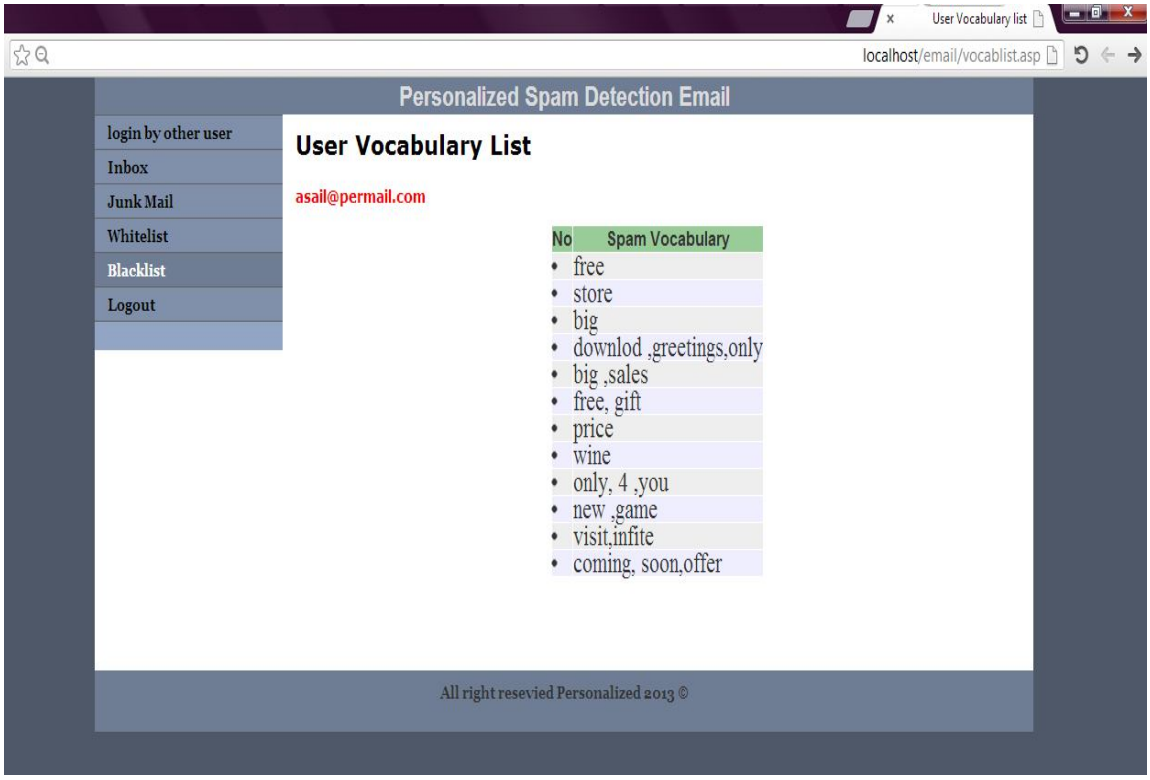


Figure D.20 User (c) vocabulary list example