AREMUCCS: A REcursive, MUlti-stage machine learning Classifier for Client-side Spam filtering

Submitted By

Kamini Simi Bajaj

Master of Computing, University of Western Sydney, Australia

Master of Statistics(Medal holder), Panjab University, India

A Thesis submitted for the total fulfilment

of the degree of

Doctor of Philosophy

School of Engineering and Mathematical Sciences

La Trobe University

Victoria, Australia

September 2020

Abstract

The upsurge of email spam is a by-product of the use of emails as the preferred medium of communication, professionally and to some extent also personally. Over the last two decades, this email spam has motivated email providers and academic researchers to contribute to this area of email spam filtering in various shapes and forms.

Server-side spam filtering is a mature technology that generally adopts a naïve, "one size fits all" approach to classifying emails as either spam or ham (i.e. not spam). However, user preferences are critical elements that must be taken into account for accurate classification based on individual requirements and tastes, including tolerance for high false positive or false negative rates: in other words, what constitutes spam is often in the "eye of the beholder". However, filtering at the client side has not caught, to any significant extent, the attention of researchers. In this thesis, a sophisticated client-side approach to spam filtering is proposed (AREMUCCS), where the classification approach is based on user preferences, and the ever-changing tactics of spammers. The algorithm presented is recursive and multi-stage, incorporating user profiling, a number of novel heuristics and approaches to classification; it can detect typical spammer anti-filtering approaches such as random string injection, using both structural and text features. Feature selection is performed using word embedding, the count vectorizer and term frequency inverse document frequency methods, before multiple classifiers - including machine learning (ML), deep learning (DL) and entropies – are applied. Across datasets ranging in size from 1,600 to 75,000 cases, including Ling-spam, PU1, CSMDC2010, Enron and TREC07, classification metrics for accuracy, precision, recall, F1 scores, ROC/AUC and PRC have been achieved in the range of 93-99.8%. Future directions for further enhancing AREMUCCS are considered.

Statement of Authorship

"Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or part from a thesis accepted for the award of any other degree or diploma. No other person's work has been used without due acknowledgment in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution."

Kamini Simi Bajaj 8 September 2020

Acknowledgement

This research could never be considered complete without acknowledging some of important people who supported and motivated me throughout my journey. First and foremost, with utmost gratitude, I cannot thank my supervisor, Prof. Paul Watters enough. This thesis could not have been completed without his support, his excellent guidance, inspiration, motivation and advice. Throughout my candidature, his support, suggestions, feedback and supervision have been extremely encouraging. During the thesis preparation, his recommendations and guidance helped me to keep going towards completion. I am deeply indebted for his support; he is the best supervisor.

Next I would like to express my gratitude to my co-supervisor, Dr A.S.M. Kayes for his help and moral support when the project overwhelmed me. His friendly attitude gave me self-confidence while writing the thesis and helped me towards completion. I cannot fail to mention support from Uncle Alex, Dr Alex Ng for inspiring me to keep moving ahead at critical times. His consistent support and care for my wellbeing helped me to complete this thesis. An important contributor to my thesis is Mrs Angelika Maag who spent numerous hours reading and editing my thesis. A big thanks to her with gratitude for correcting my English.

I would like to thank La Trobe University for giving me the opportunity to complete my research and thesis. This work was supported by an Australian Government Research Training Program Scholarship.

I am grateful to my employer Western Sydney University, especially Prof. Simeon Simoff, Dean, School of Computer, Data and Mathematical Sciences for his support and allowing me to take time off from my academic duties to focus on my research. Without this, I could not have achieved my goal of completing my PhD work. I am obliged and thankful to him. Finally, I am thankful to my family, particularly my sisters who are my pillar of strength for anything I do in life with their unbounded and unconditional love for me and my children, they gave me reason to progress in life by calling me their 'hero'. I have relied so many times on their love and that of my friends who believed in me.

Last but not least, it is my God who gives me courage to face challenges and strengths to do well in life and encouragement to complete my thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

"The only crime that has been proven is the hack. That is the story."
– Ramon Fonseca

1.1 Introduction

Emails been fully accepted as a communication medium since their origin in 1960s and the number of emails sent daily has exponentially increased every year since. The estimated number of email users are 3.8 billion and it is expected that this number will grow to 4.4 billion in 2023. The total number of emails sent and received every day worldwide in 2018 was approximately 281 billion. It is projected for 2023 that this figure may increase to over 347 billion emails exchanged daily. The main reasons emails are the most popular choice for communication are the advantages associated with their use such as immediacy, spontaneity, convenience and low cost. Furthermore, storage of data generated by email is easy and inexpensive.

However, since emails are so extensively used, they also come with problems. Together the high usage number and advantages associated with emails have led to an upsurge in email spam over two decades, increasing the total percentage of email spam-from 36% in 2002 to 96% in 2006-2007. It started declining post 2010 and has been stable at around 50-60% over the last 5 years.

A fact recognised many centuries ago is that wherever money exists, there is fraud. The motivation for this thesis is to develop a dynamic and user focused machine learning based solution to combat a particular type of fraud associated with emails called email spam, an abuse of electronic messaging systems-by violating privacy with unsolicited information, products or services.

Though it began as a form of advertising, the wide use of email has created avenues for opportunists for scams and fraud. This may be in the form of identity theft with the intention of making unauthorised withdrawals from bank accounts, gaining access to sensitive financial information in the case of companies or the use of credentials to obtain credit elsewhere. However, it may also be aimed at the disruption of systems through malware and many more (Ford & Spattord, 2007).

Email spam has been faced by everyone who has used email applications via computers, tablets or smartphones for information exchange by sending or receiving email messages from last two decades. One reason for its continuous existence is that it is profitable for spammers who, with limited investment, are able to advertise and lure users to click on unexpected link, disclose their personal data or download viruses and malware. Even at its lowest, during the 2009 economic downturn, spammers were able to rake almost $4000.00 per day from its Viagra spam campaign. Today, spammers earn $7000.00 on average per day with only 1 response received per 12.5 million email spam messages.

As discussed earlier, email spam can be of many types and may lead to other forms of cyber fraud such as phishing, DDoS attacks, botnets, viruses/malware, sniffing, key logging, and identity theft, causing billions of dollars in losses. Email spam evidently is the origin of all these other types of cyber fraud. Earlier, the latter were spread via viruses; however, with the emergence of sophisticated techniques for generating trillions of emails to reach user inboxes, easy access to has been created to also commit other forms of fraud. This often starts with embedding viruses and malware in the spam email, which is downloaded by the user. However, even without opening the spam email, significant damage can be caused. Out of the many listed, phishing and botnets are perhaps the most commonly used. Phishing coaxes users into revealing personal financial information such as bank login credentials and credit card details. Using these bank details spammers transfer funds from the user's accounts and make purchases using the illegally obtained credit card details. In many cases, the stolen credentials are sold on, causing significant financial losses for individuals and institutions. Botnets work on a different principle where a network of private interconnected devices infected with malicious software is controlled as a group without the knowledge of owners. These botnets can be used to perform distributed denial-of-service attacks, steal credentials, generate and send email spam and also allow the attacker access to device and connection. Each device in botnet may run one or more bots that are able to produce billions of emails per day. Hence, it is important to understand that controlling email spam not only saves users and organisations time, resources and money but also indirectly tackles other fraudulent activities. Finding a solution to these problems could be said to be a 'one size fits all' approach to combating cyber fraud.

Such solution needs to come from technology that has the ability to analyse vast amounts of data generated by immense email exchange. The most promising of these is machine learning (Grobelnik, Mladenić, & Fortuna), a set of algorithms that work together to improve performance through, among others, prediction capabilities. When presented with exemplified sets of labelled data, these algorithms learn and improve their ability to predict. These algorithms have been widely used by researchers and email providers in an attempt to combat email spam. However, so far no one algorithm has been proven to be superior to another, in fact many of them produce similar outcomes. The main reason is that the majority of algorithms focus only on text features contained in the email and do not account for 1. Differences in individual users' preferences, 2. structural attributes and 3. anomalies introduced into the text features by spammers.

1.2 Research Challenge and Scope

Since the early 2000s researchers, commercial organisations and email providers have been developing techniques to control the influx of email spam resulting in well-developed anti-spam filters (details are presented in Chapter 2). They are based on header analysis, list-based (i.e. blacklists, white lists, grey lists) for address and keywords, protocols, content filtering (at the mail server) with further classification through statistical analysis based on learning systems, artificial immune systems (AIS), reputation-based filters, ontologies, social networking, p2p, and grid computing (G. Caruana, Maozhen, & Man, 2011; Pelletier, Almhana, & Choulakian, 2004; Pham, Lee, Jung, and Sadeghi-Niaraki, 2011; Vu Duc & Truong Nguyen, 2012; Ying Zhang, Yang, & Liu, 2012).

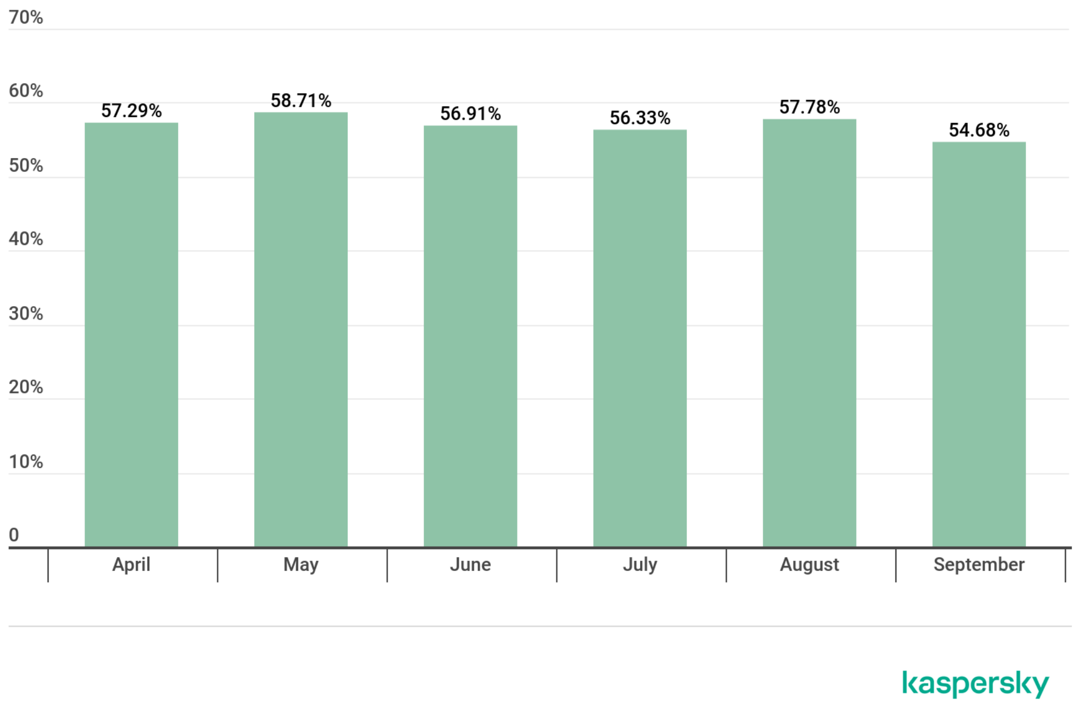


Figure 1.1: Percentage of Email Spam in Q2 and Q3 in 2019 (Vergelis, 2019)

Figure 1.1 shows the percentage of email spam from April to September 2019 where the highest number of spam in global email traffic occurred in August. Despite extensive efforts by researchers and leading IT companies, who have developed complex anti-spam software solutions, email spam has continued to grow quickly.

However, the statistics above indicate that the issue is still unresolved, and further research is needed to look for anti-email spam solutions to counteract new techniques devised by spammers. Success in controlling email spam remains elusive and is like a tug of war despite more than a decade of research efforts. Hence, the problem which is the focus of this thesis is well researched; however, we are approaching this issue from a different viewpoint.

Existing anti-email spam solutions can be categorised into prevention and detection techniques. Prevention is aimed at stopping email spam at the source while detection techniques focus on the recipient, identifying an email as spam and directing it to the spam folder. While prevention is highly desirable, this is difficult to achieve, to some extent due to the fact that spammers continue inventing new ways of circumventing the filters. Furthermore, they seize external computers to generate and send spam without the knowledge of the owners. They also often operate only intermittently. Hence detecting a source and shutting it down is almost impossible. This is further complicated by the fact that the same email, for example 'advertising a product', may be legitimate for one user and spam for another. Such scenario brings forward a challenge with the use of listing techniques as prevention. With control of the source remaining elusive, most research has focused on detection techniques.

Detection is possible at the server and the client site. In both cases, research has achieved some success with content filters, based on machine learning techniques (Balakumar & Vaidehi, 2008; Blanzieri & Bryl, 2008; Godwin Caruana & Li, 2008; Chih-Chin & Ming-Chi, 2004). At the mail server level application of content filtering solutions has the following important points to consider. First, the content filtering technique is common to the entire pool of emails received by the mail server on behalf of thousands of email users of the organization. Second, as K. Bajaj and Pieprzyk (2014) and Shajideen and Bindu (2018) point out, the classification of an email as spam is subjective to each individual user whereby it may be spam to one user and legitimate to some other user. Therefore, with an aim to filter email spam if the classifier is kept very stringent, this may result into a large number of false positives. A false positive (FP) is a legitimate email that is tagged/classified as spam, this may create false alarm and block emails coming from legitimate sources. Isacenkova and Balzarotti (2014) flagged this as significant because user missing out on receiving an email in their inbox may cause them a loss of important information. On the other hand, if the classifier is kept at a low level of filtration, it may lead to higher false negatives. A false negative (FN) is an email spam that is classified/tagged as legitimate email, indicative of the problem of email spam in the user's inbox.

A sample email spam shown in Table 1.1 from a public email spam dataset called Ling-spam first presented in J. K. Androutsopoulos, Chandrinos, K.V. Paliouras,G and Spyropoulos, C.D. (2000a) that was successfully classified as spam by a content-based filter. The content filer identified words such as ‘information’, ‘$2 million’, ‘1-800’ as spam words and classified the email as spam.

Table 1.1: Sample Email Spam with Text Features from Ling-spam dataset

```
Subject: re: information requested
hi, name is john ' m 27 years old. was able $ 2 million working
home, 'd share did. please few moments busy life listen short
message tell! call listen, 1-800 - 764-6203 change life!
```

As pointed out above, in order to overcome the content-based filters the spammers constantly find new techniques to mislead filters. Gargiulo, Penta, Picariello, and Sansone (2009) and K. Bajaj (2017) cited examples of how email spam has evolved to contain concealed words with inclusion of special characters and numbers. The word ‘sale’ may be modified as ‘s@1e’ to trick the filters. Thus, email spam now includes more than words in the form of links, numerical digits, special characters etc. These non-textual features, as shown in the sample email in Table 1.2, are often successful in misleading text-based filtering mechanisms and are not be identified as spam. Non textual features are structural features that define the size, date, time of the email or other characteristics associated with the email.

Table 1.2 Sample Email Spam with Non-Text Features from Ling-spam dataset (K. Bajaj, 2017).

```
Subject: free promotional offer
' ' own 100 % free web site site : http : / / 000000138 .
0000127 . 000044 . 00000005 . cearth . . ca / users /
freewebsites / * * * charge * * * * * * commitment * * * * * *
problem * * * opportunity s33kers internet m@rketers small lagre
site is . s1te linked thousands web sites ? amazing site . . .
http : / / 000000000138. 000027 . 44 . 5 . cearth . . ca / users
/ freewebsites / * * * charge * * * * * * commitment * * * * * *
problem * * * is tru1y going site century ! * * * * * * * * * *
* * * * * * * * * * * * please excuse intrusion . one fr33 offer
mailing * * * * * * * * * * * *
```

Another important issue that is the subject of research is concept drift, the subject of recent investigation by several researchers (K. Bajaj & Pieprzyk, 2014; Sheu, Chu, Li, & Lee, 2017), meaning a continuous change of the topics of the spam emails. The content of email spam generally focuses on certain topics that either may be of interest to users or are prevalent in a particular time period. For example, many topics of email spam in 2013 were related to Apple, Microsoft, success of Steve Jobs and Australia post (with the aim of targeting Australian users), national holidays in the US, malicious eCards, the birth of the royal baby in the UK, Edward Snowden's FBI hunt, Spain's railway accident and delivery failure notifications. The email spam topics reported by *Securelist* in 2019 were Valentine's Day, Apple Products, fake technical support, new features in Instagram, mailshot phishing, financial spam, job offers, ransomware and cryptocurrency, and malicious attacks on the corporate and banking sector. An experiment with email spam and evaluation results related to concept drift are presented in Chapter 4.

A notable feature of email spam is the size of the email as reported by Kaspersky lab. In Figure 1.2, it shows the movement in the percentage of emails of various sizes for Q1 2019 as compared to Q4 in 2018. The share of very small emails (up to 2 KB) in spam increased in 2019 to 73.98% (shown in red in Figure 1.2) against 67% shown in green in Figure 4) Q4 2018. It is evident from this data that the majority of spam emails are smaller than 2kb.

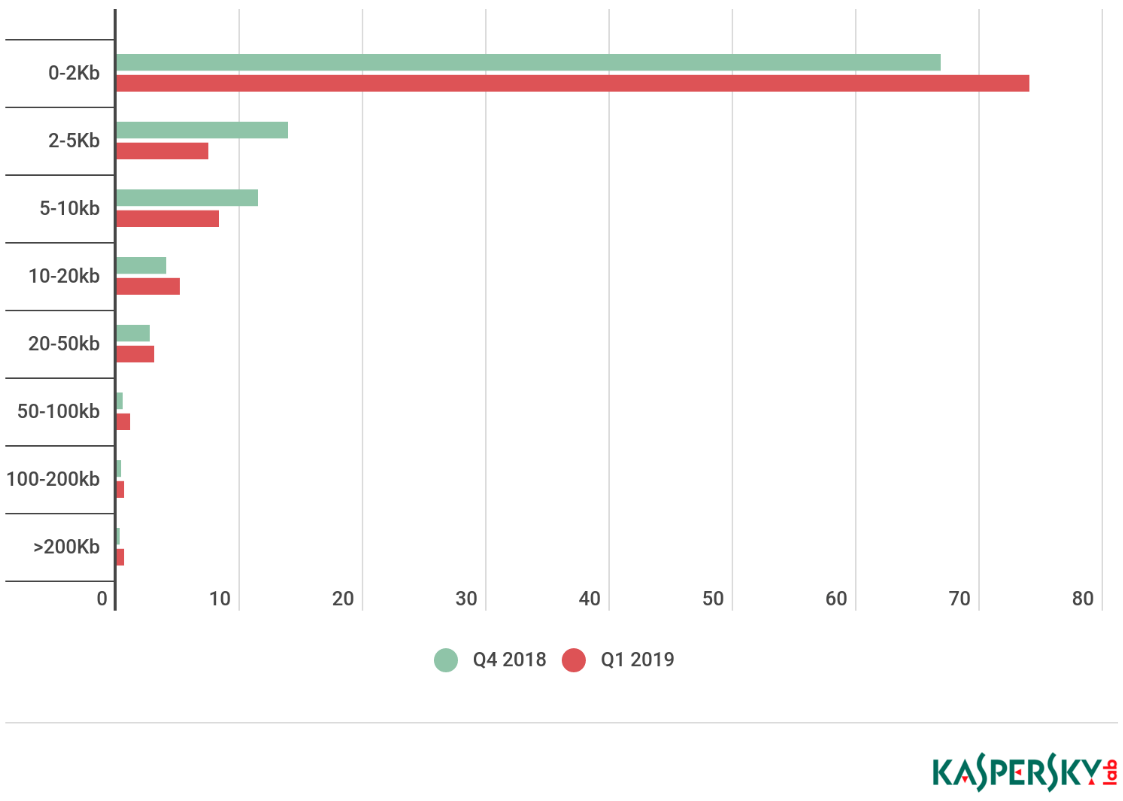


Figure 1.2 Percentage of email spam of various sizes (Vergelis, 2019)

All of these examples of non-text structural features in email spam highlight their importance in content filters. The spam reports published in the 2nd quarter of 2015 by Kaspersky labs (Shcherbakova, Vergelis, & Demidova, 2015) shows non-textual structural features for email spam filtering based on the features identified in spam emails that spammers are using to deceive the filtering solutions. Some of the examples of such features are IP address modifications, introduction of upper and lower-case letters, special characters, numerical digits and symbols, incorrectly spelt words, email size and presence of links to direct users at forged sites.

From the foregoing, it is clear that spammers confound researchers by constantly changing their tactics, that the magnitude of spam is not reducing, and that there are potentially costly misclassifications at the end user's site because user preferences are not taken into consideration.

Therefore, it is required to improvise the method of spam detection to incorporate user preferences, syntactic features and techniques to detect latest spammer techniques to make the process of spam detection efficient so that the percentage of email spam in users inbox can be reduced.

This work takes inspiration from the following research challenges:

- There is lack of an appropriate approach to incorporate individual user preferences into email classification methods. Though there are approaches available that builds models based on use behaviour with an aim to classify emails using these models, user behaviour is not the right reflection of user preferences. Typical user behaviour associated with an email involves read, save, delete, report as spam. Unless a user reports an email as spam, no other behaviour helps the model learn. How many of us either delete or do nothing when we see s spam email in our inbox? Just relaying on user behaviour is not an effective way of identifying user preferences for type of email they want to welcome in their inbox. Majority of machine learning model(s) with automated feature selection mechanism are heavily based on the nature of data used to characterise the learning of these models. Hence, user data can be utilised to model user preferences.
- There is relatively small body of publications available on client-side email spam filtering solutions. This thesis focuses on the under-developed and ignored area of client-side email spam filtering, proposing that a multi-level filtering system can improve the user email experience and minimise the loss of important emails as FP. This thesis highlights the need to reclassify emails detected as spam by a mail server. Most existing client-side email spam filtering solutions are based on a single machine learning model, although it has been demonstrated that ensembles of machine learning models outperform these. For this purpose, we develop a multiple stage filtering mechanism at the client side for reclassifying the emails identified by the first round of filtering.
- Current machine leaning models heavily rely on semantic features contained in either header or body or both which, in isolation, may not be enough to detect email spam. Structural features have equal significance in terms of patterns that can be identified. In this work, feature selection takes into consideration the structural components of an email, as syntactic features to design a more effective model for classification.
- Current techniques for spam filtering are proving inadequate against the ever-changing tactics of spammers who bring noise into emails to confuse the filters. Forged emails represent another side of the tactics which is very prevalent today and is very easy to achieve as email systems do not require authentication. This research focuses on the

development of an independent feature for machine learning models to classify emails. The main target is the type of words present in spam. They differ from those used in legitimate emails; however, spammers infuse their email content with random pieces of valid text as strings to confuse filters and lower their capacity for protection. In some cases, the random text is even scrambled to confuse the content filters. These tactics can be countermanded through entropy distribution using a series of machine learning algorithms.

With these challenges in mind, this research aims to develop a sophisticated filtering system at the client end capable of appropriately segregating emails for that individual user based on their preferences. In summary, server-side mail filtering is not sufficient to classify incoming emails correctly, and current client-side filtering does not effectively countermand current tactics of spammers.

1.3 Research Question

The broad aim of this thesis is to reduce the impact of the threat coming from the loss of important legitimate emails and the clogging of users' inboxes with email spam. More precisely, the following research question is addressed:

Are the existing machine learning based filtering solutions for email spam detection effective against the challenges posed by changing spamming techniques?

To answer this broad question, we ask:

a. Is email spam generic to all users?
b. Are current email spam control measures at the mail server level adequate?
c. Is second level email spam detection required?
d. Should the greys be manually segregated?
e. Are text features enough to detect and classify email spam?
f. Would using single machine learning technique effectively reduce False Positives?
g. Do ensemble-based machine learning techniques increase the efficiency of email spam detection via filtering?

1.4 Hypothesis and Research Goals

The overall intention of this thesis is to identify the optimum level of features to feed into machine learning model(s) to reduce the number of FN (email spam) and minimise FP (losing legitimate email as spam). This is possible through the customisation of user inboxes to individual preferences and needs. The goal is to achieve this with minimum over head of time and resources for the user.

The thesis investigates the following hypothesis:

A client-side multi-stage machine learning classifier based on text and non-text features significantly minimises FN and FP for individual users.

This hypothesis is tested through developing a recursive multi-stage classifier for email spam filtering at the client side (AREMUCCS), which first develops profiling for individual users using machine learning and deep learning models. AREMUCCS then measures the performance of a dynamic multi-level ensemble model (DMLEM) based on CART, SVM, kNN and LR machine learning models. DMLEM builds attribute sets using semantic and syntactic features as a sparse frequency matrix. AREMUCCS addresses the new tactics of spammers such as addition of random strings to mystify the filters and uses entropy distribution as email classification method based on, among others, LR, SVM, XGB, NB and RF and deep learning models such as CNN and DNN. Given the specific feature sets, feature selection is carried out using count vectoriser, term frequency inverse document frequency and word embedding methods. AREMUCCS assumes that features are conditionally independent given the class labels, though in reality there is some dependence between the features. Experiments demonstrate that the resulting model is easy to fit and works.

AREMUCCS does not require users to enter thresholds, numbers or types of features and is a fully automated algorithm. It works on original feature space; this means that the learning algorithm incorporates all knowledge acquired during its usage as original feature set not as modified or transformed feature set for classification by the machine learning models. Furthermore, AREMUCCS does not incur high computation cost that is generally assumed to be the case with learning algorithms.

Thus, there are several research goals:

1. Each user's definition of unwanted emails (i.e. spam) is tailored to their preferences and needs, to develop an effective user profile.
2. Email spam filtering at client site provides another level of filtering that trains filters according to the individual user profile.
3. To develop a fully automated system that obviates the necessity to manually classify emails. This is accomplished by an ensemble of machine learning models, evaluated for performance.
4. To develop a system for email classification incorporating techniques to counteract the latest methods deployed by spammers such as random string infusion.

5. To incorporate the deep learning models as the classification model for user profiling, topic-based classification and random string detection.

1.5 Thesis Contribution

In order to achieve these goals, the aim is to accomplish the following tasks:

1. Provide an overview of existing email spam classification techniques with close focus on machine learning based classification. A survey of mail server level solutions versus client-based solutions is conducted, with special emphasis placed on email spam detection techniques. In order to classify an email, it is important to understand how email spam filters work, especially learning-based email spam filter models. Close attention will be paid to existing feature selection and machine learning models with a view to using them for this research.
2. Propose and implement a system that develops profiles for each user based on their dataset. This profile will distinguish user preference and need from other users but is sufficiently generic to be applied to any user and be customisable.
3. Survey and select an existing client-based email spam filter to develop proof of concept for client-based email classification. Experiments will be conducted to validate that client-based user specific filtering improves performance.
4. To develop and implement a system that requires no involvement from user to manually classify emails, while providing high performance in terms of minimising rates of FP and FN.
5. Define and select semantic as well as syntactic features that comprehensively cover necessary parameters for accurate classification. These features will be used to form a feature set, used to train the model. The feature set is formed using feature selection methods: countvectoriser, tfidf and word embedding.
6. Develop an ensemble model based on the selected features, incorporating a range of machine learning models and evaluating performance on selected datasets.
7. Identify specific features and techniques to combat the latest spammer techniques such as legitimate random text or strings in spam emails that spoof the filters. The use of entropy distribution, machine learning models and deep learning models to classify as spam emails with valid random text.

1.6. Thesis Outline

The remainder of this thesis is structured as follows:

Chapter 2 introduces the key concepts and provides details on the email information life cycle, classification, modelling and filtering techniques. The current state of art in spam filtering is examined in the literature review and available work on email spam control is enumerated as taxonomy of control measures.

Chapter 3 gives an overview of classification models, datasets, supervised machine learning and deep learning algorithms used in this framework. Different models are presented and compared. The performance evaluation metrics used to evaluate the models are introduced and defined.

In Chapter 3 a typical email model is described along with elaboration of a learning email spam filter which forms the basis for machine learning algorithms. It then outlines the theory underpinning the client-side email spam filtering systems. Experimental details for client-side email spam filtering with user specific data are provided along with results. This chapter introduces a Bayesian Classifier as case study example of client-side filtering where this classifier classifies emails into three categories of spam, legitimate and greys; here greys are the emails that have the attributes of both, spam and legitimate emails hence are not clearly classified. This chapter also presents performance testing experiments of this classifier using token types, max-discriminators (token sizes) and thresholds. Results of the experiments are reported along with the discussion justifying the need for another layer to improve the performance of this Bayesian client-based classification filter into two-class output.

Chapter 4 A, MUlti-stage machine learning Classifier for Client-side Spam Filtering with sub sections on feature selection, a dynamic ensemble classifier, DMLM: Dynamic multi-layer model for high precision classification of greys from the Bayesian classifier introduced in Chapter 3 and further improvised into DMLEM using methods for combining machine learning models into a bagging ensemble. DMLEM, a Dynamic multi-layer ensemble model focuses on improving performance of the Bayesian classifier by reclassifying spam and classifying greys into spam and legitimate emails to achieve an acceptable level of FP and FN.

Chapter 5, User Profiling to incorporate user preferences for email classification is outlined, evaluated and experimental results are presented.

Chapter 6 presents email classification based on Random String detection, using entropy distribution and subject based email spam classification based on machine and deep learning models. Experimental results to validate the model are also reported in this chapter followed by a conclusion of thesis and future work in Chapter 7 which also provides analysis and the discussion. The main contribution of this thesis is covered in Chapters 3, 4, 5 and 6.

Chapter 2

Research Motivation: Email Spam as an Issue over the last 40 years and Proffered Solutions

"A love letter lost in the mail, forgotten, miss delivered and then discovered years later and received by the intended is romantic. A love letter ending up in someone's spam filter is just annoying."

— *B.J. Neblett*

2.1 Overview

The attributes which make email the preferred communication medium for exchanging messages is convenience and cost-effectiveness. The message size in an email may vary from one kilo byte to many megabytes and sending is very cheap using the email address and the simple mail transfer protocol (SMTP), a communication protocol for email transmission. One of the features of the SMTP protocol is that it allows sending messages to anyone, not dependent on any email client or provider. An undesirable consequence is email spam. The features that differentiate spam from legitimate emails are, firstly, that the email is unsolicited, and the receiver had no intentions of receiving it. Furthermore, the sender of the email is unknown to the receiver and they have no direct or indirect connection with this sender who has directed the message to a massive number of email addresses. Email spam accounted for almost 58% of all emails sent in August 2019. However, this is not a recent phenomenon. In fact, the problem existed already more than 40 years ago with the first unsolicited email sent by Gary Thuerk in 1978 to promote a new model of computer (Deffree, 2019).

2.2 Why is unsolicited email called SPAM? A historical review

The Word ‘spam’ was originally the brand name for Hormel Foods, maker of the canned "Shoulder Pork and hAM"/"SPiced hAM" luncheon meat since 1937. The term was suggested by Ken Daigneau, brother of the Vice President of Hormel Foods in a contest to win $100 to name ‘spam’. However, today the term "spam" has come to mean network abuse, particularly for unsolicited email and massive junk postings.

Historically, spam originated in the 1970s as a mechanism for advertising via messages that were sent to large numbers of recipients regardless of whether they had subscribed to the advertising message. 1978 marked the sending of the first spam message that caught attention of scientific literature. In 1982, Denning (1982) was one of the first to study the issue of email spam. However, spam in early 1980’s, also known as unsolicited commercial email (UCE), was merely an innovative way of sending information to large cross-sections of people, while today it has become a serious threat. Spam itself is controversial as the message an originator could be the means of advertising products or services or an unwanted message -a ‘nuisance’.

2.3 Email Spam as an Issue

Overall, spam is today one of the major social issues as the abuse perpetrated through electronic messaging systems includes most broadcasting media and digital delivery systems. This means that the email system may not only be used to circulate text messages but also for promoting unsolicited information, products or services, and turn an electronic tool into a relay for spam. They have become a staging ground for attacking other systems and spreading malware or indiscriminately spying to capture identity information, such as bank account details and credit card information (Ford & Spattord, 2007). The email addresses are collected in such a way that they are useful for advertisers. However, such emails are unsolicited emails and, hence, spam.

Spamming is economically viable as the only cost associated is the cost to manage the mailing list. An ‘Email Statistics Report 2013-2017’ by the Radicati Group Inc. stated that in 2013, 3.9 billion email accounts were registered from which 929 million mailboxes were for organisations. This means that more than 50% of the world’s population use email and that the average number of email accounts is 1.75 accounts per user. In the same year, most of the email traffic came from business emails, which accounted for 100 billion emails per day, a 9% increase from 2012. Overall, 190 billion emails were sent daily in 2015 (Symantec, 2016) which rose to 269 million in 2017 (Radicati, 2017), 281 billion in 2018 (Radicati, 2018), and 293.4 billion in 2019 (Radicati, 2019); this is projected to increase to more than 347 billion emails exchanged daily by 2023 with the current 3.8 billion email users worldwide are estimated to grow to 4.4 million in 2023 (Clement, 2019).

This is not without dangers, however. Online technologies make it relatively simple to disguise or misrepresent identity, or to make use of someone else's. As a result, even if the user replies to a spam email with a disguised identity, it never reaches the sender as the sender address is only temporary. Hence, it is difficult to identify the true sender to report the issue. Thus, the history of spam elimination attempt is long and troubled.

According to statistics, the percentage of emails spam sent over the Internet increased from 36% of total emails sent worldwide in 2002 (Clifford, Faigin, Bishop, & Brutch, 2003) , to 45% in 2003, 64% in 2004 (Jaeyeon & Emil, 2004), and 68.6 in 2005 (Leavitt, 2007). Siponen and Stucke (2006) conducted a study of 500 businesses in the USA and suggested that 81.6% of all email traffic was spam in 2005. It increased to 86.2% in 2006 (Leavitt, 2007), and 92.6% in 2008 (Han, Kim, Ha, & Jo, 2008). There was a small decline to 90% in 2010 (OstermanResearch, 2011) and, again, to 86% in 2011 (Gudkova & Namestnikova, 2011) as a result of the permanent elimination of the Rustock botnet in March 2011 which compromised 1.1 to 1.7 million computers. Symantec reported that spam decreased from 92% in August 2010 to 74% in October 2011. Thereafter, the statistics for email spam stayed around 71.8% in 2012 (Namestnikova, 2012) and 70% in 2013 (Patidar, Singh, & Singh, 2013). This figure has been around 50-60% since 2014. The email spam rate for 2018 was recorded as being 55% (Clement, 2019). For the first quarter of 2019 the proportion of email spam in mail traffic was reported as 56% which rose to 58.71% in May, averaging 57.64% in the second quarter of 2019.

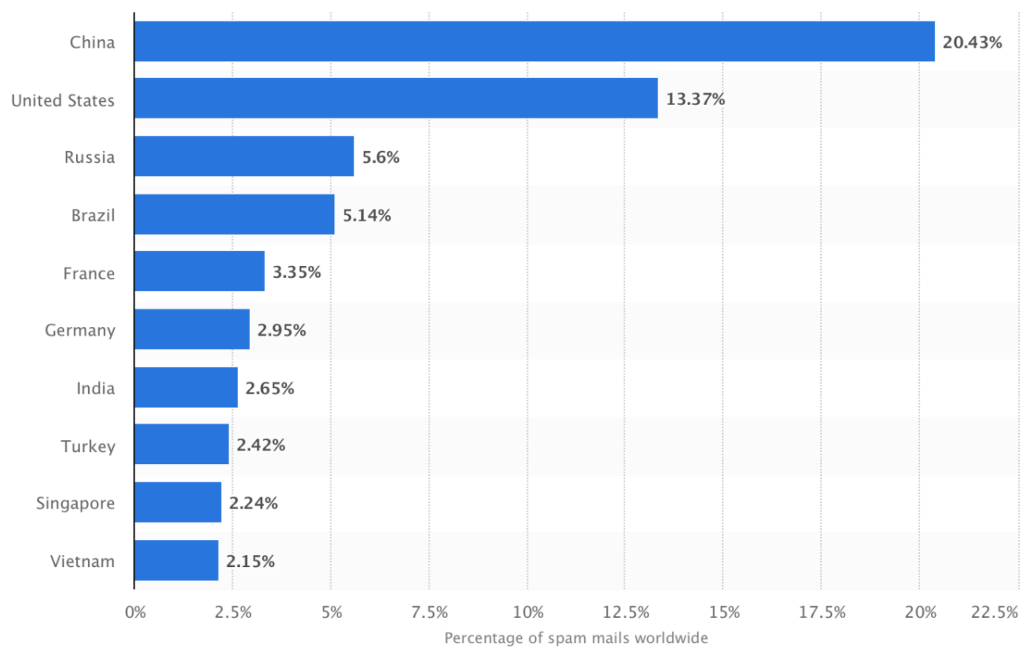


Figure 2.1: Leading countries of origin for email spam in 2019 (Statista 2020)

An increase of 20% in email spam occurred from 2005 to 2006 when it reportedly rose to almost 90% of all email traffic towards the end of 2006. Antispam vendor message labs reported it as being 89.4% from which 27% originated in the United States of America (USA) and 26% in China. Brazil, France, India, Russia, South Korea, and the UK were also significant sources of email spam in 2006. In 2019, spam from China peaked at 20.43% and from the USA at 13.37% (Figure 2.1). A report by MacAfee in 2009 revealed that in 2008 alone, 62 trillion spam messages were sent.

2.4 Impact of email spam

Email spam is a problem, clogs the inboxes of users and makes it difficult to distinguish important messages from spam hence wasting time in reading and deleting. They also consume computing and networking resources such as storage and bandwidth and are frequently used as a sinister tool for cybercrime activities such as the denial of service attack (DoS), distribution of malware, phishing, stealing sensitive information to name a few. Rao & Reiley (2012) report that approximately 90 million spam emails were sent out daily in 2010, responsible for 88-90% of the email traffic worldwide during that year.

The impact of spam can be summarised as:

1. Waste of computing resources - spam consumes bandwidth, introduces delays in routing and causes unnecessary processing (Dada & Joseph, 2018a). Some businesses store spam messages for analysis to find a solution (Pour & Kholghi, 2012) which further adds to wastage of computing resources.
2. Loss of productivity - clearing mailboxes from unwanted emails wastes time and can be highly frustrating for the recipients (Dada & Joseph, 2018a).
3. Denial of service - flooding networks with spam can create bottlenecks and blockages making communication via the network impossible (Blanzieri & Bryl, 2008).
4. Invasion of Privacy - collection of addresses from email recipients may be carried out without their knowledge - an invasion of privacy that then also exposes senders to further unsolicited emails.
5. Fraud and Deception - popular kinds of spam that create the perception of offering users' financial opportunities (i.e., 'get rich quick schemes' a Nigerian Letter scam). Other methods may prompt users to provide access to confidential information (i.e. their email account or bank details).
6. Unnecessary expenditure – cost may be incurred for deploying anti-spam systems that process and delete email spam (OstermanResearch, 2011) .
7. Identity theft – the main intent of some spam is to collect private and sensitive information to get access to valid credentials (OECD & Ahn, 2004).
8. Spreading Malware – email spam can also be dangerous in other ways as it may contain viruses, trojans or the kind which may damage software and systems (Ali Elsiddig, Elhadi, & Ahmed, 2017).

9. Security breaches – some spam causes security breaches in systems and networks (Sanz, Gómez Hidalgo, & Cortizo Pérez, 2008).

10 Loss of reputation – forged emails are sent with an intention of damaging the reputation of an individual or an organisation

11 Bounced emails due to forged return addresses - undelivered emails are returned to the sender by the mail server with notification causing network clogging.

12. Reduced consumer confidence –financial and information losses may lead to loss of trust in digital technology and emails among consumers.

From the above, it is evident that the problem of email spam is economically quantifiable. In a press release in 2009, Gartner stated that more than 5 billion US consumers experienced financial losses through phishing attacks, a particular type of email spam - 39.8% more than the year before (Gartner, 2009).

In summary, three main goals of spam were identified: 1) Gain illegal financial advantage 2) extract data and 3) compromise networks and systems. Within a few years of its introduction, email spam morphed from nuisance to serious security risk to individual and organisational data; therefore, this needs further research (Ali Elsiddig et al., 2017; K. Bajaj & Pieprzyk, 2014; S. K. Bajaj & Pieprzyk, 2013). According to Ferris Research, which studied messaging and content control, the cost of email spam to organizations in the USA was USD 8.9 billion in 2002 with a 12% increase in 2003 to $10 billion and to $17 billion in 2005 (Ferris_Research, 2007). This rose to $20 billion in 2006 (Leavitt, 2007) and 2011 (Rao & Reiley, 2012). In 2010, email spam cost global economies around $130 billion in combating 107 trillion email spam sent as reported by the World Economic Forum in 2016 (Smith, 2016).

In Japan, the amount of lost GDP was about 500 billion yen in 2008 (Takemura & Ebara, 2008), while according to a report by the government of United Kingdom, about 1.9 million incidents from email spam cost citizens an estimated amount of £10 billion in 2016. ACMA investigated compliance through the Spam Act and reported the number of spam complaints received for Jan 2013 to be 51,525 from which 50,477 were complaints related to emails. The Behavioural Science Lead from MWR Info Security, Adam Sheehan, reported to The Economic Times in 2018 that email spam had gained traction as a successful attack vector with click rates rising to 14.2% from 13.4% in 2017. In Australia, a 2019 report by the Australian Competition and Consumer Commission (ACCC) Scamwatch showed the number of reports as 1,67,798 and financial losses as $142,934,416.00 (Figure 2.2).

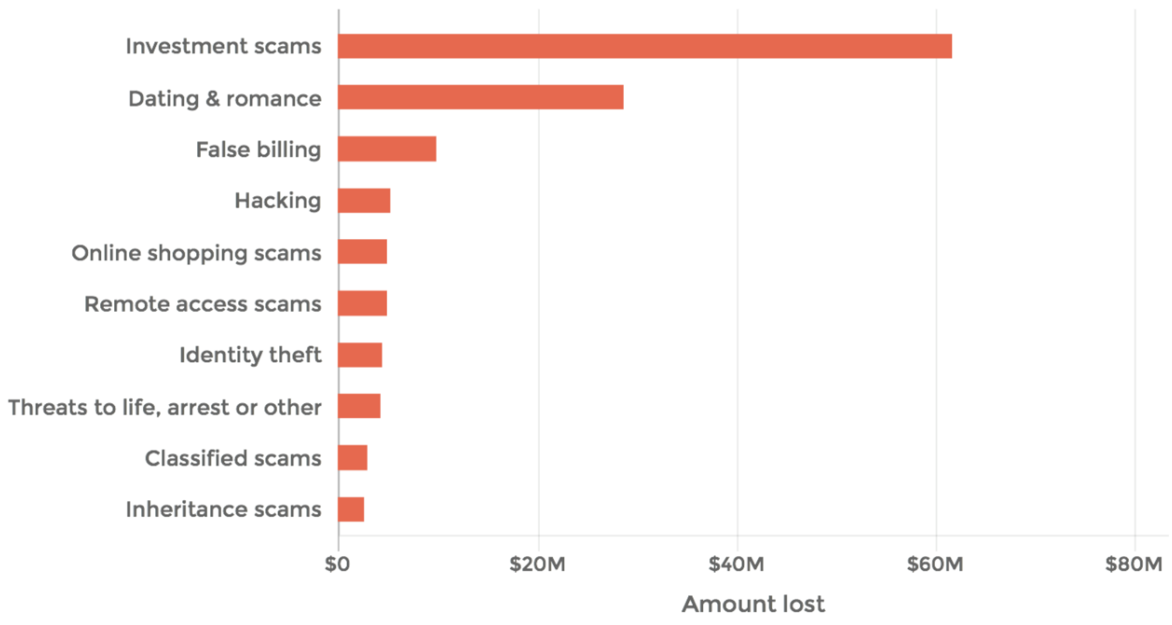


Figure 2.2: Amount Lost for Spam Types from ACCC Scamwatch

Dhinakaran, Chae, and Lee (2007) have argued that this impacts on the existence and popularity of emails (i.e. low despatch cost and ease of sending it through various software tools) (Dhinakaran, Chae, & Lee, 2007; Takumi et al., 2007). High financial outlays by consumers and organisations globally (Deepak & Sandeep, 2005) counteract the potential benefits of emails (Figure 2.3). When to this, the negative impact of wasted computing resources, loss of productivity, denial of service, invasion of privacy, as well as fraud and deception are added, some doubt could be raised about the viability of email communication (Ferris Research, 2007; Nagamalai et al., 2007; Toit & Kruger, 2012). Furthermore, the impact of these undesirable consequences may still increase as email use is still growing.

2.5 Categories of Email Spam

Email spam can be grouped into categories based on different criteria. Most are applicable to Networked PCs, laptops and mobile devices. Many researchers have studied the content of spam messages for developing taxonomies (Aradhye, Myers, & Herson, 2005; Balakumar & Vaidehi, 2008; Chao & Yiming, 2007; Chih-Chin & Ming-Chi, 2004; Drucker, Donghui, & Vapnik, 1999; M. R. Islam, Wanlei, & Chowdhury, 2008; R. Islam & Wanlei, 2007; Kun-Lun, Kai, Hou-Kuan, & Sheng-Feng, 2002; Zhen, Xiangfei, Weiran, & Jun, 2006). This section investigates email spam based on content and attachments. Some authors have categorised email spam according to the type of content in the spam, popular families of email spam or genres (Sanz et al., 2008). Taxonomies/types of email spam are discussed in the next section.

Email spam can be characterised into two major categories

2.5.1 Spam without Attachments

These are email spam messages containing text messages with or without a clickable URL to a web source but no attachment. It can be further classified as content and link email spam:

2.5.1.1 Content Email Spam (Jindal & Liu, 2007)

This kind of spam contains text messages as means of advertising, marketing or scam. It may content text that may lead the reader to take an action based on the information in the email. In many cases. it does not serve any particular purpose other than annoying the reader. One classic example is the Nigerian letters scam. Most of the sender's mail accounts do not exist. Such emails are limited in number a when compared to other categories. In the case of text messaging spam in mobile devices as SMS, the spammers target users with lucrative offers and ask them to respond whereby the cost of sending that message is high, leading to financial loss to the users.

2.5.1.2 Link Email Spam (Guoyang et al., 2006; Jindal & Liu, 2007)

Contemporary search engines rank web pages by taking into consideration the number of links connected to the pages. A web page to which more links (called in-links) are connected is more likely to be ranked higher. Spammers, therefore, often attempt to manipulate links on the web by, for example, adding thousands or even millions of links to the pages they want to promote – a technique referred to as 'link spam'(Guoyang et al., 2006). Link spam in emails is where an email contains a link either to an external source to advertising or an attempt to discover the identity of the individual via a phishing attempt. The link in the email may lead to downloading viruses and malware on the individual's device.

Such types of email spam use techniques to alter the logical view of the content of the email such as keyword stuffing, hidden text inclusion, or doorway links. The overall look and feel of such emails generally convince readers of the purpose of the email and they may take the intended action. On mobile devices, link spam on the messages prompts the user to click on a link which would take them to another website to complete the task. This may lead to the user device acting as a spambot generating numerous messages beyond its capacity and, thereby, clogging the device preventing it from its intended tasks or compromising user identity.

2.5.2 Spam with Attachment

This kind of spam contains combination of image, text and URL or a clickable link to a website. The attachments in most cases are image or executable files that may be embedded in or attached to an email. The image files are mostly in .gif format.

2.5.2.1 Image Spam

Image spam has been growing since the early 2000s and is now a serious problem as text-based filters can analyse the textual components of an email. However, for image spam the textual content is implanted into images attached or embedded into the email body. The origin of image spam comes from the limitations of the conventional spam blocking tools that rely on textual analysis of incoming messages which does not work well against image spam. The volume of image spam has increased dramatically from <4% in 2005 to over 40% in 2007. By 2010, around 85% of the email spam contained images (Bhowmick & Hazarika, 2018). In a March 2007, a survey conducted by Osterman Research reported that more than 60% of messaging decision makers cited image spam as a problem for their organizations. There are various combinations of image spam: Image and text as attachment, Image, text and URL and Image and URL (Dhinakaran et al., 2007; Sirisanyalak & Somit, 2007). Image-based filtering has been reviewed in detail by (Ching-Tung, Kwang-Ting, Qiang, & Yi-Leh, 2005).

2.5.2.2 Executable file (Virus Spam)(Qiu, Hao, & Chen, 2004)

Spam with executable files as an attachment is intended to spread viruses and try to establish mail bombs to plan DDoS attacks on the mail servers and networks. Upon execution of the attachment, the machine acts as a zombie and performs tasks intended by the spammer such as downloading big programs to harm the network or automatic generation of emails to others in the same domain clogging the entire network (Dhinakaran et al., 2007).

2.6 Types of Email Spam

A taxonomy of spam in terms of needs to include advertisements for products and services (such as adult content spam, pharmacy and medicine, software, personal finance, education), phishing, denial of service attacks and distribution of viruses and malware. Due to the commercial purpose, this is denoted as UCE (unsolicited commercial email) and is seen by organisations as a tool to approach potential customers because email is a low cost and convenient way to reach large groups of people. Furthermore, email spam is represented as the following types:

- Commercial Advertising
- Scam
- Phishing
- Financial
- Image spam
- Botnet spam
- DDoS

  Malware

Pour and Kholghi (2012) mention two more types as word obfuscation and backscatter spam.

2.7 Taxonomy of Email Spam Control Measures

Basic anti-spam techniques and measures try to separate spam from legitimate emails. The literature includes several research works that propose techniques for combating email spam, including blacklists, whitelists, grey lists, content-based filtering, feature selection methods, bag-of-words, machine learning techniques such as Naïve Bayes (Vu Duc & Truong Nguyen, 2012; Ying Zhang et al., 2012), Support vector machines (G. Caruana et al., 2011), and neural networks (du Toit & Kruger, 2012). Further, there are lazy learning and reputation-based techniques (Zheleva, Kolcz, & Getoor, 2008), artificial immune systems (Xiao-wei & Zhong-feng, 2012), protocol-based procedures, and many more that have not been listed here (Aldwairi & Flaifel, 2012; Balakumar & Vaidehi, 2008; Pham et al., 2011; Xiao, Junyong, & Meijuan, 2010); (du Toit & Kruger, 2012; Horie & Neville, 2008; Huai-bin, Ying, & Zhen, 2005; M. Islam & Zhou, 2007; Klonowski & Strumiński, 2008; P. Liu, Dong, & Zhao, 2007; McGibney & Botvich, 2007; Moon, Shon, Seo, Kim, & Seo, 2004; Nhung & Phuong, 2007; Rajendran & Pandey, 2012; So Young & Shin Gak, 2008; Wei, Feng, Di, & Feng, 2010; Wu & Tsai, 2008; Xiao et al., 2010; Xiao-wei & Zhong-feng, 2012; Ying Zhang et al., 2012). Godwin Caruana and Li (2012) list some emerging approaches such as peer to peer and grid computing, social networks and ontology-based semantics along with several other approaches.

Many authors have conducted surveys and listed measures to control spam over the last decade (Aggarwal, 2012; Blanzieri & Bryl, 2008; Godwin Caruana & Li, 2008; Lai, 2007; Nazirova, 2011; Paswan, Bala, & Aghila, 2012; Quinten, van de Meent, & Pras, 2007).

These solutions can be grouped into categories such as list-based and filtering techniques. A different categorization proffered by Nakulas, Ekonomou, Kourtesi, Fotis, and Zoulias (2009) is by purpose (i.e. prevention, detection and reaction). Paswan et al. (2012) categorize email spam filtering techniques as origin-based, content-based or traffic-based filtering, and as feature selection or feature extraction methods. This thesis has analysed these categories and has broadly classified spam solutions into three categories: Regulatory Laws and Legislatives, Education and Awareness and Technical Measures, with the major focus being on technical measures.

2.7.1 Regulatory Laws and Legislation

The economic damage and violation of laws caused by email spam have resulted in some countries in the implementation of new regulations and legislation (Guzella & Caminhas, 2009; Stern, 2008). Efforts have been made by individuals and organizations such as governments, ISPs, anti-spam organizations, consumer protection organizations, and organizations providing anti-spam solutions at commercial as well as non-commercial level. In a global network such as the internet, resisting email spam requires uniform global legislations and compels the creation of uniform universal laws. An article published by an Organization for Economic Co-operation and Development, France, provided a list of 39 national and international anti-spam, mostly non-commercial, organizations (OECD & Ahn, 2004). The following legislative measures have been set up by a large number of countries using two kinds of approaches – (a) existing laws and regulations which, though not specifically addressing spam, may nevertheless be implicated by some aspects of spam, e.g. laws to protect consumers from deceptive marketing or to prevent the distribution of pornographic images and (b) amendment of existing laws and regulations or creation of new regulations to address the problem of spam (OECD & Ahn, 2004). There is a range of regulatory approaches such as opt-in, opt-out, ISP rights and responsibilities, scope of spam (for example the US CAN SPAM Act of 2003 allows for UCE but places restrictions on it), spam ware, disclosure of personal data, EU member states as well as National Cyber Alert systems and complaint mechanisms (Blanzieri & Bryl, 2008). In order to avoid spam in Australia, the Spam Act 2003 sets out the responsibilities under the Australian Law. It was reported that 80% of spam in Europe and North America originates from fewer than 200 spammers operating illegally (Hoanca, 2006).

The laws and legislation addressing mobile devices are: General guidelines provided by US-CERT for mobile devices such as securing the device, posting (sharing) the device number and email address carefully. The user should refrain from following links sent in emails or text messages, being wary of downloadable software and applying security settings, all of which is also applicable to networked PCs (McDowell, 2006). The US Federal Government CAN SPAM Act for mobile devices prohibits sending unsolicited commercial email messages to wireless devices without prior permission. The Commission found that SMS messages transmitted solely to phone numbers (as opposed to those sent to addresses with references to Internet domains) are not covered by these protections.

Unfortunately, a code of conduct provides only limited protection against "bad" spammers. Spammers easily find methods to side-step systems and studies show that the Acts have had little impact on spammer activities and the number of spam emails sent (Grimes, 2007; Kigerl, 2009; Nazirova, 2011; Schryen, 2007).

2.7.2 Education and Awareness

Educating spam victims may play an important part in reducing spam. Awareness could turn those victims who unknowingly post their email addresses on public sites into spam free users. This also would increase the efforts spammers have to put in to collect email addresses. Such social approaches cannot eliminate spam; however, in general they increase awareness about spam and the way to deal with it. Legal provisions can control the problem to some extent although steps taken by informed users will certainly help reduce the problem if not eliminate it. Consumer protection and government organizations have raised public awareness by informing consumers about spamming tactics and providing them with suggestions on how to protect against them

(OECD & Ahn, 2004). Examples come from the National Cyber Alert Systems by the US Computer Emergency Readiness team that published a document on ‘Defending Cell Phones and PDAs Against Attack’ (McDowell, 2006). Furthermore, the US Federal Trade Commission operates a website dedicated to spam awareness.

2.7.3 Technical Measures

There are several measures used to prevent and detect spam; however, among all anti-spam measures used for combating email spam technical measures have proven to be the most effective.

These control techniques can be classified as techniques to prevent, detect or react to email spam. Some of these approaches are included in this section.

Technical measures are broadly categorized into protocol based (Mir & Banday, 2010). filtering techniques which can be further sub-divided into learning-based , P2P computing, grid computing, ontology-based approaches and social networks (Godwin Caruana & Li, 2012).

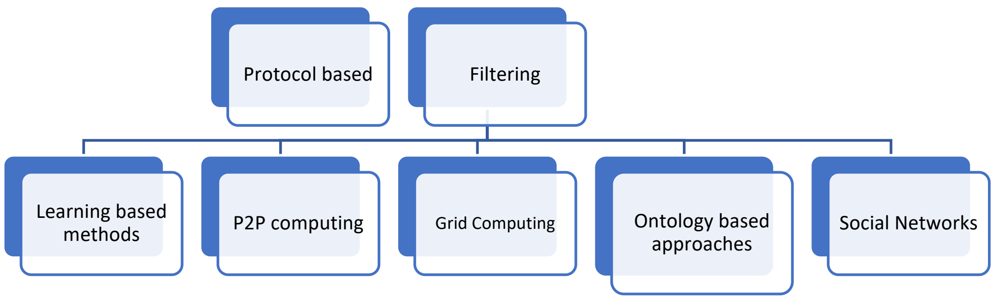


Figure 2.3 Technical Anti-Spam Measures

The focus of this research is learning-based filters for spam filtering, which can be implemented using content-based methods such as statistical, rule-based, machine learning and a mix of these. These methods use the features identified from the selected components of the spam email for classification as spam or non-spam. Since focus of this research is around learning-based methods, the following will elaborate leading up to content-based filters which is the main focus of this research.

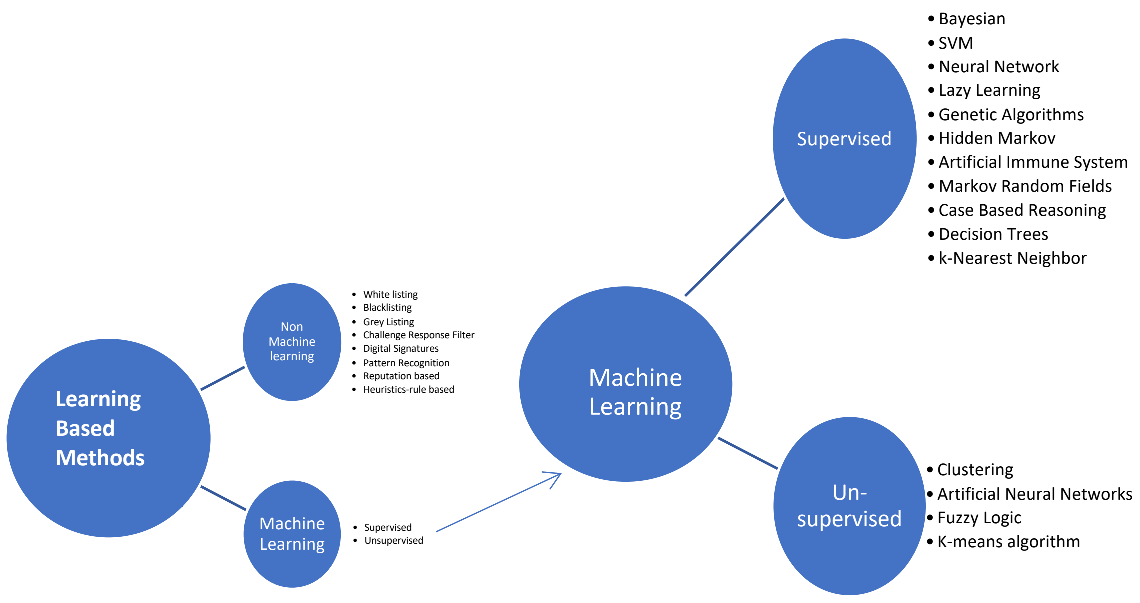


Figure 2.4: Learning-Based Spam Control Measures

2.7.3.1 Non-Machine Learning methods

Whitelist/Blacklist Filters

Whitelist/Blacklist filtering as a preventative technique which forms part of integrated filtering systems and is one of the most popular approaches to email spam filtering. Most web filters (parental controls) have incoming and outgoing packets, go through a filter driver and use a blacklist/whitelist approach.

White-lists (set of email addresses of users whose messages are allowed) and blacklists (email or IP addresses known to be spammers) are used to filter spam by using the e-mail address, IP address and DNS address. Real-time Blacklists are one kind of application based on this method. (L. Yang, Bin-Xing, & Li, 2006). However, lists are vulnerable to address spoofing and may also include legitimate messages from users who are not in a white list or who are present in a black list by mistake (Garg, Battiti, & Cascella, 2006).

A Whitelist is used in instant messaging networks and contains a list of email addresses of people that are allowed to email and are therefore on a known list of good email addresses while other addresses are blocked (Pour & Kholghi, 2012). These lists contain email addresses of people permitted to email while others are blocked.

This assumes that everyone in the whitelist is a genuine non-spam source and, without bias, would probably not become one in future. Again, there are several open issues here. Firstly, this completely discounts that individuals may become spammers and unknown email addresses would be denied access. Although some individuals may not find this problematic, companies, organisations and businesses will as they depend on as yet unknown customers and clients to remain functional.

Thus, while for non-organisational users the whitelist is relatively difficult to circumvent by spammers, as it primarily deals with known identities and gives users total control; thus, unsolicited and unidentified emails are reduced or avoided. However, a change of identity of whitelist members will also lead to automatic rejection. This means known users need to be aware that they must first communicate the change of identity before they can use their new identity which can be problematic if they are on a large number of whitelists. Although his may still be manageable, in the case of organisations where users get emails from a large number of people, it becomes a problem to build and maintain a whitelist.

In contrast, blacklist filters are access control mechanisms that maintain a list that will allow in messages and email addresses unless these are on the list of 'bad' sources; in other words, it maintains a list of email addresses that are not allowed. A well-maintained blacklist will result in zero false positives although, when implemented on a wider scale, it may be difficult to maintain, and some genuine emails may be eliminated in the process of excluding spammers. However, the effect on spammers may be severe enough for them to deem it unprofitable to spam.

Greylisting

This method rejects emails on the assumption that legitimate mail transfer agents will always retry while spam ware will not. Instead, it will go on to another spamming message on its list. To initiate their attacks, spammers use spam ware like rat ware rather than normal mail transfer agents; and if the receiver's network or internet service provider identifies the characteristics of spam ware as different from legitimate mail transfer agents, it will reject or flag the mail as spam in a short message transmission protocol session (Bhowmick & Hazarika, 2018).

But grey listing has been found to delay mail from new IP addresses on the theory that it is a spam source and even if it retries, it will be on blacklist before the mail can be accepted (Levine, 2005). Thus, while grey listing is a simple, effective way to weed out undesirable messages with few mistakes in detection, it can fail by losing legitimate mails. This can be an issue but as grey lists can be bypassed by automatically resending the message (Pour & Kholghi, 2012), this minimises the problem. However, the delay is also an issue, because delaying delivery of a particular email can be problematic and therefore may not be suitable for a company, business or organisation.

Digital Signatures

Digital signatures, also known as fingerprints, identify messages via this detection technique. In some cases where secure data is involved, messages without digital signatures are identified as spam. Signatures of messages that have been identified as spam can be put in a database. This database is then used to compare the signature of received emails with the list of signatures of spam. If there is a match, the email is spam (Pelletier et al., 2004; A. B. M. S. Ali & Xiang, 2007).

Messages coming through an unsecured channel with a digital signature indicates to the recipient that the message was sent by the claimed sender. The signature can be provided by the sender or the service provider.

Challenge/Response Filter (CR)

This type of prevention technique is often referred as ‘Handshake’ or ‘Turing Test’ (Nakulas et al., 2009) and may automatically prompt the user or sender of an email to access a website to validate their authenticity. There are two entities to CR systems, a secret value and a variable response value. Passwords, CAPCHA- a computer system used to differentiate human from machine input and biometric techniques for authentication are examples of CR systems (Muhammad Iqbal, Muneeb Abid, Ahmad, & Khurshid, 2016). This may be likened to a permission filter which blocks all emails not coming from recognised, authorised sources. Thus, emails sent to an address that uses a permission filter will result in an auto-response inviting or directing the sender to go to a web page for opt-in information and their email address will be whitelisted as authorised for future emails (Bhowmick & Hazarika, 2018).

CR uses spam filtering with dynamically updated URL statistics, a measure which calculates the probability of whether an email message is spam or legitimate based on statistical analysis. A real world deployment of such challenge response system was implemented and evaluated by (Isacenkova & Balzarotti, 2011).

Challenge-Response systems send a challenge to potential (as yet unknown) senders requiring them to perform an action so that their original message can be delivered. Once they receive a confirming email of validation, they will automatically be white listed. Although this method may be criticised for wasting time, it appears to achieve relatively positive results in confronting spammers and reducing unwanted emails. It ensures a measure of authenticity and security which cannot be compromised.

Pattern Detection

Detection methods identify the patterns in emails based on graph theory, data analysis, clustering and operations research. Identified commonalities in a large sample of emails is used to identify a pattern matched to incoming emails to calculate a score for the message (Dada et al., 2019). The higher the number of patterns identified in an email, the higher the score. Based on this score, it is classified as spam or legitimate email. This technique is also called ‘rule-based’ or ‘heuristics-based’ email spam filtering and is one of the most advanced techniques used for email spam detection over last few years (Muhammad Iqbal et al., 2016).

Reputation Filtering

Reputation-based filtering systems can be regarded as modern identity-based spam filtering techniques because they make decisions based on comprehensive information about the source of a message, blocking spam rigorously and reducing false positives. This technique is of high importance to the effective performance of modern simple mail transmission protocol (SMTP) servers used for email transmission and enhances their ability to stop spammers by validating, verifying and analysing messages and source content on networks to establish trust relationships.

According to Antonakakis, Perdisci, Dagon, Lee, and Feamster (2010), this approach can be classified into two categories, of which the first is pre-acceptance filtering prior to acceptance of a message (i.e. IP reputation). Another category is post-acceptance filtering after the message has been accepted (i.e. content-based signatures). However, the effectiveness of these techniques varies based on the type of SMTP sender, a client who is sending the email. The emphasis is on pre-acceptance anti-spam technique filters of the sender, essential in checking spam because SMTP servers and custom IP reputation lists constitute 90% of sources used by all spam senders; however, effective IP reputation filtering can significantly reduce the load on email delivery systems and reduce the exploit of legitimate SMTP servers by spammers for spam messages (Esquivel, Akella, & Mori, 2010).

Collaborative Filtering

Collaborative filtering is a detection technique based on a distributive approach where collaborative information is shared as knowledge base for the community (Sophos, 2013). Since email spam is sent to massive numbers of users, it is likely that the same email has been received by others. Some organizations prefer tagging/identifying large numbers of messages sent as spam. In collaborative filtering, an email is tagged as spam for all similar users when a particular user marks an email as spam (Beigy, 2012).They are either labelled as spam and sent to the inbox or to a spam mail box. Such identification is applied to large numbers of users and does not take into account the contents of the email, but is based on collaboratively identified information (Bhowmick & Hazarika, 2018). This kind of centralised spam filtering is more economical than the personal approach; however it may prove to be more costly in the sense of misclassification and reclassification of incorrectly classified email spam (Nazirova, 2011).

Community Filters

With community filters, users scan emails automatically for spam; and if discovered to be spam, its characteristics will be reported to a central server held at the database. Once the email spam has been reported enough times by enough people, it will be automatically blocked or filtered out for other users in the future (Deepak & Sandeep, 2005). It can be adapted by users who prefer selective blocking of spam mail based on their interests; users can also identify nuisance mail, mark it as spam and block it from getting to others in future. This is a positive technique that prevents particular spam from replicating and prevents future unsolicited emails. The disadvantage is that some users will be the first to receive the spam message which is initially stored on their database, wasting their bandwidth. There are possibilities of false positives as one person's idea of spam may be different from another, which may lead to blocking legitimate emails from legitimate senders.

Payment-At-Risk

Spammers send millions of bulky spam messages daily using considerable bandwidth at significant cost to internet and network service providers. This cost has to be passed on to the user in form of internet service bills. It could also result in reduced efficiency on the part of internet or network provider. Furthermore, precious time is wasted in clicking or checking spam by the receiver whereas it costs the sender or spammer little.

For example, if the cost to an Internet service provider is \$2 per gigabyte of transferred data and a typical spam message is 3K in size, 333,000 spam messages cost \$2 in bandwidth. Considering that spammers spam in bulk, they could send about 432,000 per day, that is, 13.4 million spam emails per month. As reported by Spam Laws, this figure is currently 10 billion (conservative figure) per day, and the cost to an internet service provider might be as high as \$30,000 in bandwidth per day. Furthermore, if it takes roughly 5 seconds to stop any activity to check or delete spam, 10 billion spam per day cost the world 50 billion seconds or 1585 years in lost productivity per day.[1]

Spammers are insensitive to the consequences of their activities and need to be dissuaded by being made to pay by the internet service providers for the waste of bandwidth occupied by unwanted spam blocking servers. This would be a feasible deterrent to reduce spam. To execute this, all service providers must act in unison and agree to force spammers to pay for the inconvenience of the spam and the server clean-up. Kuipers, Liu, Gautam, and Gouda (2005) proposed a zmail 'zero-sum' email protocol which requires the sender of email to pay small amounts called 'e-penny' to the receiver of the email. Mail senders who require to send emails within their established average will not pay or profit from this email set up with their email service providers, but spammers will have to pay significant amounts because they send out in bulk. Such a technique would moderate their behaviour. The authors regard Zmail to be an accounting relationship among complaint email service providers to reconcile payments to and from users which can be implemented on existing short message transfer protocols.

Limit rate

A reactive technique which is also a preventive measure is limiting the number of emails sent by a user, thus reducing the rate of emails sent. This technique can mainly be used at the ISP or mail server level. This rate limitation is irrespective of a user having been identified as spammer or not (Nakulas et al., 2009). Another limiting mechanism comes from Kholghi, Roudsari, and Pour (2011) in the form of a counter-based filter which saves time as the mail server may decide the legitimacy of an email before it is received. However, the limitation is that legitimate emails may not pass through this filter.

[1] http://www.prismemail.com/abouteconomics.php

2.7.3.2 Content-Based Filtering - Machine Learning Methods

The most common email spam detection technique is content-based filtering. These filters can be built manually or automatically and are traditional types of filters that have been in existence for a long time and have been effective and relatively reliable. They analyse contents of an email message body, the message subject and the email headers searching for clues that indicate spam.

Machine learning, one of the sub-fields of computer science, aims at developing self-learning systems that automatically improve their performance based on new knowledge gained. It explores building models from a given set of emails labelled as legitimate (non-spam or ham) treated as negative and non-legitimate (spam or unwanted) that are used for filtering incoming emails. Once the models are developed, they need to be trained before they can perform the filtering function (Pour & Kholghi, 2012). Email spam filters based on machine learning methods retrain themselves based on the experience and have been reviewed in the past with notable contributions from a range of researchers (J. K. Androutsopoulos, Chandrinos, K.V. Paliouras,G & Spyropoulos, C.D. , 2000b; Blanzieri & Bryl, 2008; Cormack & Cruz, 2009; Cormack & Lynam, 2007; De, Irani, & Pu, 2013; Graham, 2002; Guzella & Caminhas, 2009; Mojdeh & Cormack, 2010).

A typical email consists of components such as the header, the body and attachments. The models based on machine learning algorithms that classify emails may use different features of the mail to make decision about them. Email spam can be identified based upon parameters such as header, body text, links (URLs) in email, non-content features, text fields, type of terms in the text fields or subject terms. The taxonomy based on text fields is body, title, meta tag, anchor and URL spam. Gyöngyi, Zoltán, Garcia, and Hector (2005) present the type of text terms taxonomy as repeating, dumping, weaving, or phrase stitching. The subject of terms included in the email spam is unlimited, but the most common terms found are impacted by the latest occurring topics and events in the world and change from time to time.

A survey of content-based email spam filtering techniques for the period of 2002-2019 identified that machine learning methods used for content-based filtering are Neural Network Algorithm (A. B. M. S. Ali & Xiang, 2007; Sirisanyalak & Somit, 2007), Boosting, Bayesian Statistics (A. B. M. S. Ali & Xiang, 2007; Sirisanyalak & Somit, 2007), Heuristics (Ming, Yunchun & Eei, 2007; (A. B. M. S. Ali & Xiang, 2007), Signature-based analysis, k-Nearest Neighbour, Decision Trees, Support Vector machines (Guoyang et al., 2006; Ming, Yunchun, & Wei, 2007), Visualization, Instance-based Learning (A. B. M. S. Ali & Xiang, 2007), Markov Random Field, Random Forest (Goel, 2017; Yiu, 2019), Ontology-based Machine Learning Approach (Brewer, Thirumalai, Gomadam, & Kang Li, 2006; Shajideen & Bindu, 2018), Deep learning (Barushka & Hajek, 2018; Diale, Celik, & Van Der Walt, 2019; Jacovi, Sar Shalom, & Goldberg, 2018) and Ensembles (Anandita, Yadav, Paliwal, Kumar, & Tripathi, 2017; K. S. Bajaj, 2016; George & Vinod, 2015; Singh & Batra, 2018; S. K. Trivedi & Dey, 2013; Shrawan Kumar Trivedi & Dey, 2014; W. Wang, 2010; Z. Yang, Nie, Xu, & Guo, 2006).

There has been a growing need for an efficient anti-spam filter to block all spam without blocking legitimate emails. The Bayesian filter has satisfied this need to some extent. The rise of Bayesian filtering is owed to Paul Graham, a Harvard computer science PhD. He presented Bayesian filtering as a spam filtering technique in his paper ‘A Plan for Spam’ in 2002 (Graham, 2002) which was followed by a Bayesian Filtering implementation thereafter. The Bayesian anti-spam filter has been content-based and self-learning (adaptive) in nature (Deshpande, Erbacher, & Harris, 2007).

This implies they ‘train’ from known ‘good’ and ‘bad’ emails and during training extract ‘tokens’ (separate words) and store them in a database. They return impressive detection rates with fewer false positives and false negatives. It does not require pre-set rules and analysis of message content. Under the principle of calculating a spam message, if the probability value is higher than the set of threshold, the message is classified and treated as spam and with its self-adapting technique, its degree of accuracy increases with learning (J. Kim, Chung, & Choi, 2007).

Chhabra, Yerazunis, and Siefkes (2004) presented a Markov Random Field model-based approach to filtering email spam. This approach examines the importance of the neighbourhood relationship (MRF cliques) among words in an email message for the purpose of spam classification. Guoqing, Wei, Haixia, and Jianshe Dong (2006) proposed a multi-agent based collaborative peer-to-peer system to combine a content-based filter with a P2P collaborative filter, so the system can respond to new spam rapidly as well as take advantage of prior spam knowledge.

An ontology assisted parallel scheme for scalable SVM training is presented by G. Caruana et al. (2011), utilising a distributed computing framework that uses Hadoop implementation and spam ontologies for improving accuracy. However, the use of such distributed computing techniques on the training set causes variable but noticeable amounts of degradation in accuracy.

J. Kim et al. (2007) deployed a Naive Bayes model to focus on URLs in the email messages to develop a model for the filter instead of considering words. The filter is later updated with the incoming messages that were classified. The filter is fed back only periodically for correctly classified messages but more regularly for incorrectly classified ones. The advantage this filter offers is that it is automated unlike other URL-based filtering approaches though the performance is similar.

Wu and Tsai (2008) presented a novel behaviour-based method for email spam filtering by designing and implementing a back propagation neural network. This research analysed and extracted the features from the behaviour of spammers from the headers and system logs of the emails. A classification model was developed and evaluated. From a temporal perspective, behavioural features are robust when compared to text-based features, and studies have shown that this method is more robust than textual feature-based classification.

A comparison between spam filtering and ontology-based filtering was drawn in Shajideen and Bindu, (2018), who argued that conventional filtering offers no control to users over emails, while the ontology-based filters develop users profile and propose email spam detection based on user preferences for classification, using a support vector machine.

Awad et al. (2011) reviewed the applicability of the most popular six machine learning methods on the problem of email spam classification. The algorithm descriptions of the methods (Bayesian classification, k-NN, ANNs, SVMs, Artificial immune system and Rough sets) and results of the experimental conducted on the Spam Assassin dataset are presented.

Huang and Xu (2013) proposed a hybrid spam filtering model based on user feedback identifying social network relationships among users. Identity-based and content-based characteristics of emails were utilised to develop the Bayesian model for identification of email spam, which is updated dynamically from the knowledge gained from user feedback. Results show that the model outperforms the classification performance of traditional filtering methods when the email characteristics change.

B. Zhou, Yao, and Luo (2014) present a cost sensitive three-way email spam filtering system to reduce the chances of misclassification by adding a 'suspect' folder where users can examine the suspicious emails. The model addresses two issues, the threshold values for the three classes and the interpretation of the cost sensitive characteristics of filtering problem. It computes the threshold based on a decision-theoretic rough set model and cost as a loss function selecting the one with minimum loss.

A. Wijaya and Bisri, (2016) investigated a hybrid combination of Logistic Regression (LR) false negative threshold and decision tree (DT) to detect spam emails. DT has a tendency to be oversensitive to noisy data; hence the method utilises LR for reducing noise in the data by filtering the correct prediction with false negative threshold before the data is fed to DT to build the classification model. The proposed method is evaluated using a Spambase dataset of size 4601 with 2788 non-spam and 1813 spam messages at a 61-39 ratio.

Sheu et al. (2017) proposed an efficient systematic email spam filtering mechanism based on a decision tree machine learning method to track concept shifts. The method analyses the header of the emails to identify features to detect a concept shift in new emails to classify them as spam or ham. The filtering method is incremental in nature, thereby strengthening the ability of the method to adapt to dynamic environments. C4.5 decision tree is used as data mining algorithm to support numerical as well as categorial attributes, selected for training and testing from the TREC data set, with a size of 59,116 for this method, Dada and Joseph (2018a) applied a Logistic Model Tree machine learning algorithm to develop a classification model to filter email spam. The study aimed at achieving higher accuracy using this model with a small feature set size and a sample of 5180 emails from the publicly available Enron dataset for training and testing. The classification results showed 99.3% accuracy and high false and true positive rates.

Yan Zhang, Liu, and Yao (2019) proposed a Game theoretic rough sets model for classifying emails into three categories - spam, ham and suspicious with a trade-off between accuracy and coverage. The study aims to minimise the classification error, examines the game formulation and repetition learning mechanisms of the model and compares it to the Pawlak rough sets model for email spam filtering. The authors compare and evaluate the two models using training and testing data sets obtained from the UCI Spambase dataset.

Email spam has become a real threat to the extent that many content-based filters are not able to efficiently and effectively identify it even with recent techniques; and spammers are able to obscure their intentions with confused texts, phrases, and words (Sheu et al., 2017). New spammer methods have even worse impact. They try to circumvent filters through web roots and spambots which are capable of performing human activities such as registering user accounts, browsing pages and posting web content. These filters are unable to detect them as they are trained to identify content-based spam features and not spambots. This raises a new wave of concern regarding the continuous threat posed by spammers in this “cat and mouse” game with them.

2.8 Evaluative Analysis of Proposed Solutions

This chapter identifies different types of anti–spam techniques either using filters and other characteristics to deter spammers. Although these anti-spam techniques may be suitable for some users, they may be unsuitable for others, they may achieve some level of protection against unwanted email messages. They have been implemented at commercial as well as non-commercial level.

Each of these anti-spam techniques has unique features that distinguish it from others. Although none of these may be completely able to stop all real-time potential spam since spammers continuously develop new tricks to deceive the filters, some well-designed filters achieve 90-95% success under certain conditions. However, a perfect anti-spam filter would be a combination of, firstly most unique features described and the features lacking. Most anti-spam solutions include multiple techniques such as white and blacklists, content analysis, authentication, as well as heuristic rule-based and network-based techniques. However, determining that an email message is spam is difficult through a purely automated process. Moreover, user involvement-based filters such as challenge response systems, put an extra burden on the senders of legitimate emails. Additionally, with such systems, there is a challenge where automated response systems are involved, with difficulty in answering the challenge response filter.

The main problem with many of the techniques suggested is the performance of the filters (Dada & Joseph, 2018a). There is need to increase the accuracy of classification especially with the evolving nature of email spam and the new techniques spammers apply. There is a need to find the optimum balance between performance, robustness and cost.

The problem with list-based techniques is that people can have more than one email address or use more than one location to have a different IP address. So, a new IP address or email address does not give assurance that the person is sending a message for the first time. It is usual for individuals to send an email from a new location or email address; hence filtering emails based on unknown credentials would risk losing some important emails. Another issue with lists is that it is susceptible to penetration and demands frequent updating or fails (Sheu et al., 2017).

It has been demonstrated that content-based filtering techniques are able to identify the features in the email spam and classify them as spam, using machine learning techniques. However, the new waves of spambots deceive these filters. Spammers devise new and more complicated techniques with confused words and phrases that evade the filters (Sheu et al., 2017). Moreover, the filters are trained on certain common messages but are not comprehensive; therefore, self-learning takes time for maximum filtering performance. Social interactions attract email traffic coming in form of telemarketing and opt-ins, and most existing spam filters are exposed to high false positives and false negatives. Spammers hide email contents or forge fields in the email header in order to deceive content-based filters. These undesirable methods of spammers to obfuscate and circumvent contents of filters with words or confusing terms like "via@gra", "L/0/a/n/s", "mort.gage" etc, sometimes tend to make content filters less efficient and evade detection. A sizable number of existing email spam filters are unable to keep spam away from users' inboxes.

Chapter 3 discusses the issue of email spam filtering for an individual user at the client side. It proposes a Bayesian classifier based on Naïve Bayes machine learning method to prove the hypothesis that user preferences play an important role to reduce email spam for users. Further presented are the experiments that this research aims to utilize in order to circumvent email spam at the client end.

Chapter 3

Client-side User Specific Email Spam filtering - Can we CAN the Email Spam for specific users

"The difference between a stranger sending you a message that you might be interested in at a very low volume level, no repetition, just sending it to very few people, and that being done as spam - those things get close enough that you want to be careful never to filter out something that's legitimate." ~ Bill Gates

3.1 Overview

Email spam problems can be tackled at three stages: Prevention, Detection and Reaction. Prevention is stopping the email spam before it is delivered to the recipient, to control the distribution of email spam altogether (Gupta, Kumar, & Mohandas, 2011). Some of the preventive methods are legislative controls, protocols for senders and blocking of email spam senders (Nazirova, 2011). Detection means identification of an email received as spam, at the destination and reaction is the response after an email has been detected as spam. It would be an ideal solution to have the techniques for prevention implemented to minimize email spam; however, prevention alone is not feasible as it may lead to loss of important emails due to the following reasons: the nature of this problem is diverse, the content of email spam is dynamic, email spam is subjective to recipients and there are chances of misclassification. Hence, most of the research for this problem has focused on detection as a solution for email spam reduction. In most cases, especially where supervised machine learning is used, detection is followed by reaction. Initial parts of this chapter explain email classification and filtering process where firstly, the life cycle of email is explained followed by email transmission and filtering in Sections 3.2. Section 3.3 explains learning-based filter models for email spam classification. A discussion of filtering at the server-side versus the client side is given in Section 3.4, followed by a discussion on spammers' reactivity to spam detection mechanisms in Section 3.5. The focus of this chapter is user-based email spam filtering at the client side that addresses user preferences.

It is considered a fact that email spam classification is specific to users and investigations have been carried out to test this hypothesis in Section 3.6. Here, an experiment is presented based on context specific datasets to enhance filtering performance at the client side for a user.

Some client-based filters are set at threshold cut off values for ham and spam in order to reduce the amount of email spam in users' inboxes; however, with these threshold scores, only the clear cases of spam are directed to spam folder. With this approach, a substantial amount of emails lies between the two threshold values and all emails with spam scores within those values fall into the 'grey' which requires users' involvement to manually segregate them as spam and ham. This contributes to the enhanced learning of the filter; however, it is not cost effective. In this chapter experiments are carried out with Bayesian classifier that revealed that the model with 0.5 thresholds for spam and ham each lead to a sizeable number of false positives. Feature size plays a significant role in classifier performance. The experiments are aimed at determining the optimum number of features and threshold values for a better spam detection rate by the classifier. The details of these experiments and results are presented in Section 3.8 and the analysis and discussion is provided in Section 3.9, followed by the conclusion.

3.2 Email classification and Spam Filter

In order to understand the problem of email spam, it is useful to document the lifecycle of an email since along this life cycle, there are many steps where an email may become spam. The life cycle of an email spam (Ridzuan, Potdar, & Talevski, 2010) runs parallel to the life cycle of an email as shown in Figure 3.1.

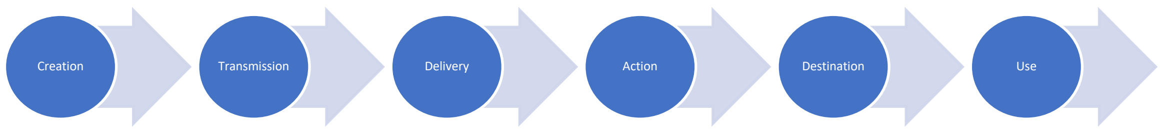


Figure 3.1: Email Life Cycle

It begins as soon as the email is created. The intention behind the ‘creation’ of an email represents the line of demarcation for whether the email should be classified as spam or ham. This newly created email is then addressed to recipient(s), sent, transmitted (‘transmission’ stage) through the network and delivered (‘delivery’ stage) to the listed recipient(s).

Once delivered the ‘action’ is that the email is analysed and classified as of interest or not. Here, the terms ‘of interest’ and ‘not of interest’ are being used for the following reasons: Initial creation of each email is legitimate and is targeting a recipient. If the content of the email is not what the recipient would like to receive, the recipient is not interested in receiving the email and, hence, the email is labelled as non-legitimate or spam. If interested, the email is classified as ham and the destination is the inbox where it goes into ‘use’ stage. If not of interest, the email can be classified as spam at any of two levels - the server or the client side. In either of the case, the ‘destination’ is the junk mail folder. ‘Use’ stage for email spam is its usage as training sample for spam email. However, some email spam bypasses the filters and stays in the users’ inbox. This may also attribute to another interesting classification called ‘grey’ - also referred to as ‘unsure’. This classification is associated with those emails which have characteristics of both spam and ham but cannot be clearly be classified as either, hence the terms ‘unsure’ or ‘grey’.

3.2.1 Email Transmission and Spam Identification

This section discusses the email transmission process and the location of spam identification. The World Wide Web or internet is used to transmit email(s) from one user to another or to a group of users. People located in geographically distant places are communicating over emails transmitted via the internet. The mechanism of operation of a typical email communication and transmission process between one sender and one receiver which is graphically represented and described with Figure 3.2.

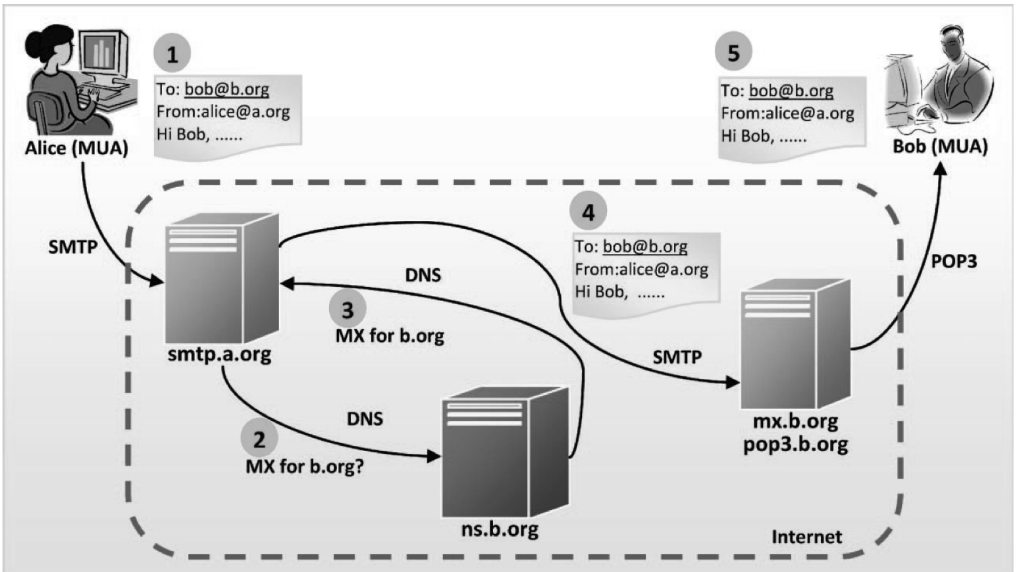


Figure 3.2: Email Transmission Process

User A, in this case Alice, sends an email to user B, Bob. The process starts when Alice's mail user agent (MUA) composes an email to Bob. The email is transferred by the email client of the sender to the SMTP server of the sender with her mail transfer agent (MTA) using the SMTP protocol (Nakulas et al., 2009). The SMTP server tries to establish a connection with the recipient SMTP server through their MTA by performing an MX record lookup though the DNS namespace service. Bob's DNS server responds with the MX record and a connection is established between the two mail servers. The email is transmitted to Bob's SMTP which then delivers the email to Bob's mailbox via the MUA of the email client. The process described above can be used for multiple emails sent and received between multiple users. For example, a series of email exchanges between 2 or more users on the same subject line is termed a 'thread' where user A is sending emails to multiple users B, C, D and so on. Another scenario is when multiple users are sending an email to user B.

Spam identification can be carried out at several stages in this email transmission process. At the receiver's (in the example, it is Bob) mail server (SMTP), or at the sender's mail server. It can also take place at the email client of the receiver. Figure 3.3 shows the life cycle of email spam where the intent of creation of an email message is to send spam, here email spam detection can be carried out at steps 4, 5 and 6.

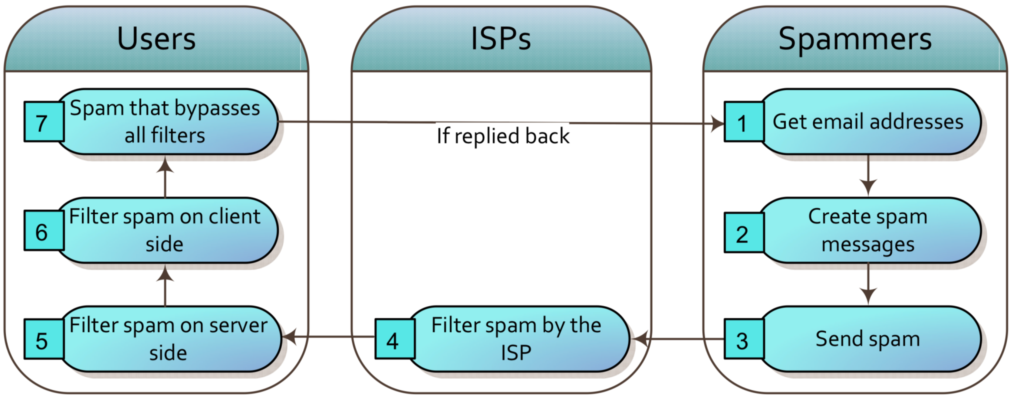


Figure 3.3: Email spam Life Cycle

Filtering at each of these levels is discussed further.

3.2.2 Email Spam Filter

Content filtering for email spam detection appears to be the most effective technique for defending users against spammers. Content filtering of emails generally falls into two categories - header and body.

An email is divided mainly into three parts: header, body and attachment. The algorithms that classify emails may use different features of the mail components to make decisions about them. The header contains structured sets of fields carrying names with specific meaning, which are displayed in a particular email varying among mail servers. The header contains information such as ‘From’ (sender), ‘To’ (recipient), ‘Subject’, ‘Date received’, ‘Date sent’, ‘routing information’, etc. The body contains the actual content of the email - either in plain text or html format. The body contains text, the URL, images or all of them. The majority of the email spam body contains text or/and a URL due to its size and network bandwidth. The third part is the attachment, which is any file type attached to the email. For example, filename, filetype. File type can be .exe, .pdf, .doc etc. The literature suggests filtering based on the header only (Khamis, Mohd Foozy, Aziz, & Rahim, 2020; Khater, 2012), the body and subject only (Saeedian & Beigy, 2009; A. K. Sharma & Yadav, 2015; Xiao et al., 2010), the body only (Ali Elsiddig et al., 2017), the body, subject and attachment (Firte, Lemnaru, & Potolea, 2010), and finally, the body, header and attachment (Fdez-Riverola, Iglesias, DÃaz, MÃ©ndez, & Corchado, 2007). The most common method uses the subject and the body for email spam detection. In this research, the focus is on the header, body and attachment. Although attachments are part of the email, the content of the attachments are not included in the text classification of the email. So, this research is limited to the type of the attachment to be included in the classification.

Figure 3.4 shows the structure of a typical email spam filter. The contents from the email message require appropriate pre-processing steps such as feature extraction and selection before they can be utilized by a classifier in a filter. The entire process of a spam filter for classification is set out below:

(1) Tokenization: All the words in the email (body as well as header) are extracted as Tokens;
(2) Lemmatization: Tokens are reduced to their root forms (for example, ‘‘retaining” to ‘‘retain”);
(3) Removal of Stop-word; words that often appear in sentences are removed from the list of tokens (for example, ‘‘as”, ‘‘the”);
(4) Representation: selected Tokens from the email are converted to a format that is acceptable to the classifier, as feature vector in Figure 3.4.

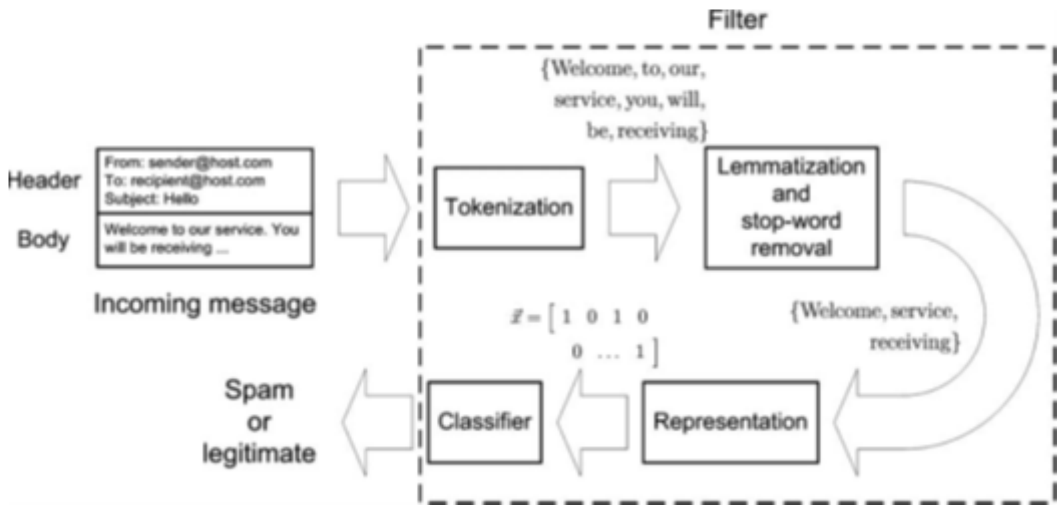

Filter
Header
Body
From: sender@host.com
To: recipient@host.com
Subject: Hello
Welcome to our service. You will be receiving ...
Incoming message
{Welcome, to, our, service, you, will, be, receiving}
Tokenization
Lemmatization and stop-word removal
{Welcome, service, receiving}
x = [1 0 1 0 0 ... 1]
Representation
Classifier
Spam or legitimate

Figure 3.4 : Structure of Typical Spam Filter (Guzella & Caminhas, 2009)

It is important to note that the process shows the analysis of the text only and represents the words as tokens. Several other components present in an email message such as links and html are identified as tokens only. In the URL, the link it points to is not included in the scope of content filtering. Pictures in the body of an email are not included in this classification but is part of image spam classification. Furthermore, it is noted that some email spam filtering approaches do not follow all of these four steps. For example, some classifiers do not lemmatize or remove stop words, while others work on raw message tokens.

3.3 Learning-Based Filter Model

In several cases email spam filtering as one of the detection mechanisms is followed by a reaction to prevent receiving the same email spam repeatedly or to improvise the detection capability and performance. Such filters are also called Learning Filters. A simple representation of the spam filtering architecture is shown in Figure 3.5 where an initial collection of spam and ham emails go through the transformation process followed by feature extraction and selection to train the model for classification (Yu & Xu, 2008). The last step applies machine learning algorithms for classification of an email as spam or ham and the decision is fed back into the model for learning purpose.

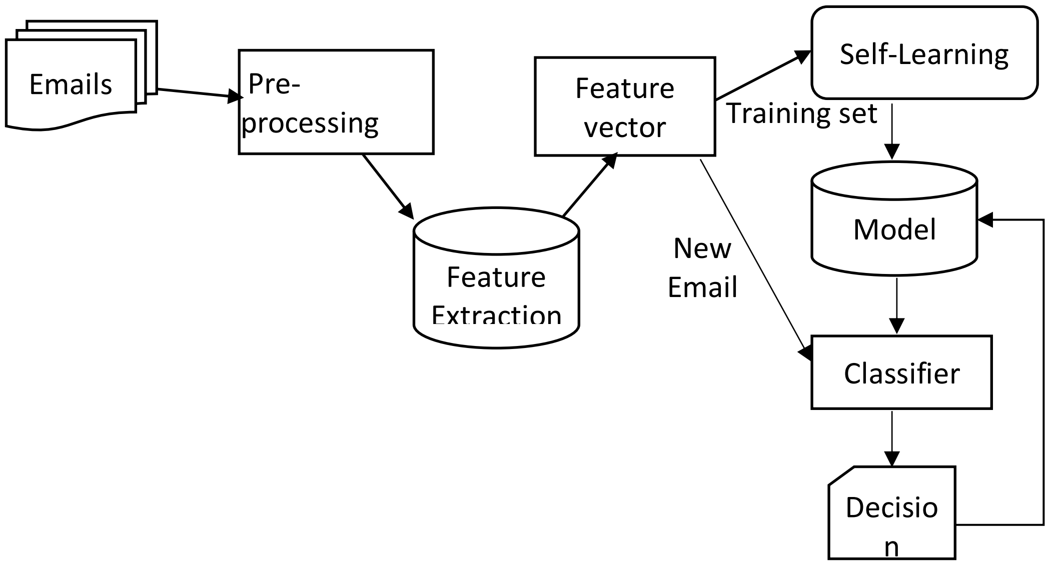


Figure 3.5: Process of Learning email filter

In general, an email filter classifies an email as either spam or ham based on the function f. This function contains some pre-defined parameters *P* that are used to classify an email *e*. The function is represented as

$$f\,(e, P) = \{e_s,\ \textit{email is classified as spam};\ e_h,\ \textit{email is classified as ham}\}$$

Machine learning algorithms for classification are a classic example of learning-based filters, and are most commonly used in filters for classifying emails (Nazirova, 2011). Machine learning, a subset of artificial intelligence enables systems to learn from data, identify patterns and make predictions without any explicit instructions and requires only minimal human intervention. It is an adaptive approach that responds to the actions taken by users of a system to improve accuracy of automated actions over time.

The strength of these algorithms is their learning capability that divides the email classification into two stages: learning stage and detection stage as shown in the Figure 3.6, also known as training and testing stage respectively (Patidar et al., 2013). The major focus of machine learning since its origin over two decades ago has been to comprehend concepts and then generalize them into models based on that conceptual learning (Thornton, 1992). These generalized set of models contain predetermined output classes that are associated with terms and concepts. A new input is classified into an output class based on features identified from it. In many cases, new input may not already have been encountered in the data used for the learning process due to the novelty of an item and its absence from existing data; however, the generalization process performs the classification based on comparable features between the new input and an output. This classification technique also deals with any discrepancies that may have arisen in the new input data

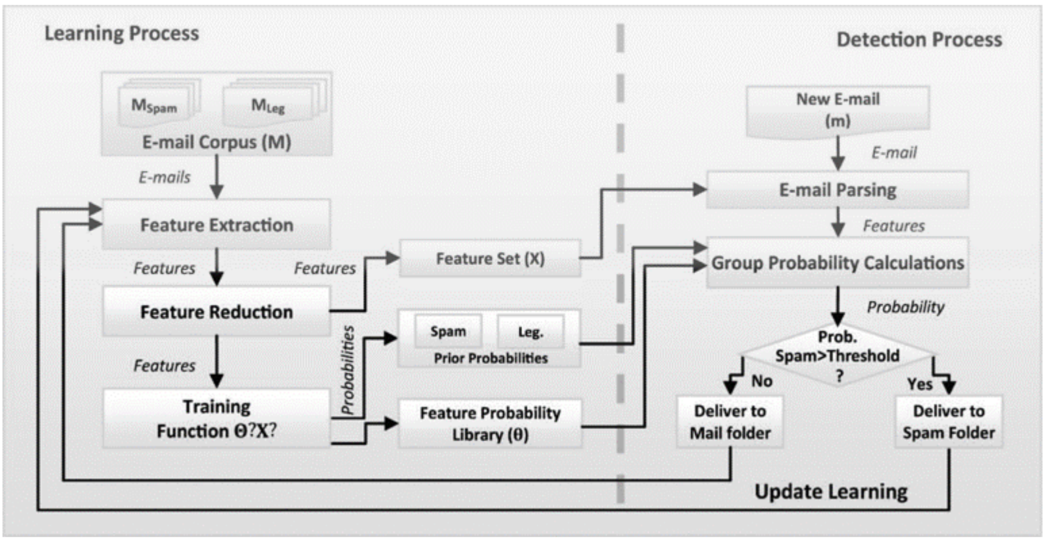


Figure 3.6: Learning email filter model (Mir & Banday, 2010)

It is worthwhile to address the two stages shown in Figure 3.6. In the learning stage, a training email corpus *M* that contains labelled spam and ham messages is fed into the filter. Machine learning improves the accuracy of classification through algorithms designed to solve the problem being addressed. For these algorithms to function as programmed, they require clean and accurate data of an optimal size, representative of the entire population it is representing. Hence, the pre-processing steps of feature extraction and feature reduction are carried out on this training dataset. A feature set *X* and a set of parameters *P* are identified and used by the training function to calculate the probability of each feature in the feature set *X* as being spam or ham. This prior probability is stored in the feature set library for later use. Hence, the training model provides the feature set with collection of features and their associated probabilities of occurring in spam and ham messages. The feature set from the library is then used for the classification task in the detection stage when a new email message arrives at the filter.

The characteristic of all forms of artificial intelligence is the capability to learn how to classify objects into one of the Z classes, by allocating input vectors to a class, based on learning from training sets belonging to each of the Z classes.

The overall goal in a classification task is to learn mapping from an input set a to an output set b, where b ∈ {s,t,….z}, with z being the number of classes. When z = 2, the classification is binary; however, where b ∈ {s,t}; If z > 2, it is multiclass classification which is also the case when the attributes of class labels are overlapping. In general, the term classification refers to multiclass classification with a single output class. In contrast, if function f represents the mapping between objects a and b shown as b=f(a), where f gives an output b for a value of a initially learned from the input dataset and later, in another instance for a new value of a that does not belong to the input dataset, f is able to define a label b. This process if called function approximation. The goal here is to generalize the function f developed from the input training set a_i that enables this function to make correct predictions on unknown input belonging to an output class.

In the email domain, the problem is assumed to be binary with two output labels as either spam or legitimate, although other class labels are also possible. The classification function can determine which class a new email belongs to depending on features or characteristics, pointing to one of the classes that have been identified through the training dataset.

In the detection stage, when a new (test) email *m* arrives, it goes through the stages of pre-processing via email parsing (tokenization, lemmatization etc.) and a feature set is deduced. Each feature identified as a result is then allocated the probability of occurring in spam and ham messages. The probability distribution P(b|a) for each class label b given the input vector a for the training set T. If spam email is represented as 1 and legitimate email as 0 then b∈(1,0). In this case, the conditional probability of email spam can be represented as P(b=1|a,T) and legitimate email as P(b=0|a,T). Based on this, a new email can be classified using the approximation function $\tilde{f}$(a)=P(b=i|a,T) where i=1,0 which can be conditioned for each machine learning model to be used for prediction. Once done for all the features identified, the total probabilities of each group i.e. spam, and ham is calculated. These total probabilities are then used to classify the email as e_s or e_h on the basis of pre-defined threshold values defined for spam and ham. If the spam threshold value is more than the total of the spam group value, the email is labelled as ham e_h otherwise it is labelled as spam e_s. Accordingly, the email is then delivered to a mail folder (inbox) or spam folder. Next is the reaction stage, where the feature probability library of the filter is updated (Update Learning step in Figure 3.6) on the basis of this classification. This learning improves the capability of the email spam filter.

3.4 Mail Server-Based verses Client-Based Email Spam Filtering

Bayesian filters have been the most effective by far and most of the email spam filtering solutions incorporate Bayesian algorithms. Initially these were utilized as stand-alone solutions; however, with the increase in complexity of the email spam problem, they are now commonly used as part of several techniques and applied in many layers. Given the fact that email spam has taken many shapes and forms, it is difficult to detect and filter email spam with only one layer. To achieve an effective level of reduction in email spam, many layers of filtering are suggested and used. There is, thus. a need for a high rate of filtering of email spam, and multi-layer solutions have been suggested in the literature. However, most of these are at the server level (M. Islam & Zhou, 2007). Multi-layer filtering can be understood as being applied at each side - email server and email client.

Nevertheless, multi-layer solutions that have been suggested at the server side have invariably given some promising results (Fahad, 2015; M. R. Islam et al., 2008; Khater, 2012); (Nagamalai, Dhinakaran, & Lee, 2007)

Server-based and client-based filtering are methods to detect email spam at the destination, a post acceptance system where emails are first received before any filtering is performed. Most popular spam filtering is performed at mail server level. In fact, the majority of the literature refers to email spam filtering as filtering at the server side only. Protecting emails at a server level is generally more beneficial for the organizations, though individual filters for the user may have their own advantages. For example, if an individual user is attacked, a filter at client level can prevent such messages being received - hence help solve the problem. However, not many filters give an option of blocking the spam emails at the email client and few studies suggest filtering at the client level (users email program) even for large providers such as Gmail, outlook or yahoo (Dada & Joseph, 2018a). There exist commercial anti-spam filters that users can install on their email client; however, they are limited and require substantial user involvement.

Server-based filtering is applied at the mail server level and is popular as it is a better choice for organizations in most cases for the following advantages. Firstly, it is easier to implement. Organization employ experts who control the implementation of the content filtering solution. Secondly, it is cost effective in two ways; it is a one-off cost for detecting email spam as early as possible rather than delivering it to clients and individual users and also, the resources required to implement the filter for the organizational users as a whole requires the same resources such as software, hardware and human resources whereas the cost involved in directly protecting individual users are high due to the purchasing cost and the maintenance of the software for the users. Thirdly, having a server-based solution allows the administrators higher levels of control since individual users may not use it appropriately, hence causing errors. The tuning of filter control is managed at one point which gives organizations more control. Fourthly, server-based solutions provide a common point of control for ensuring organizational policies for email content use are followed (Pelletier et al., 2004). Lastly, server-side solutions protect the users of an entire organization.

If server-side filtering were the best solution, then users would not face the problem of email spam; it would have minimized the email spam problem. However, spam has different meaning to different users. The same email may be spam for one user and useful for another. It is important that users have control over what emails they want to receive. Inadvertently, server-side solutions ignore the fact that users would like to choose which email messages they would like to receive and which not. Some users may find the problem of email spam bigger than others. These users may have individual preferences in relation to which emails they would want to receive and which not. The organisational filters at server level are unable to take care of such user preferences. Hence, such emails may end up in user inboxes as spam.

Receiving too many emails as spam may have substantial impact on productivity and levels of frustration. Dealing with a large number of email spam, users may mistakenly click on the wrong link in an email causing them and the organization financial loss as well as damage to reputation. It is acceptable for organizations to monitor the content of emails received by users; however, removing emails based on their content is not acceptable and should not be exercised. The main objective of filtering is to combat unwanted emails not censorship. Hence, client-based filtering that gives users rights to control which messages they want to receive is important and can offer higher accuracy in filtering assuming appropriate user data is available to guide the filters (Kolcz, Bond, & Sargent, 2006).

Client-side filtering is applied directly at the client's email program as an add-on and is usually more accurate. It provides users with options in terms of deciding what kind of email they want to receive in their inbox and may reduce the total number of email messages they receive as spam. Client-based filtering provides protection for user emails on an individual level. In client-based filtering, close attention is paid to users' personal information. Email spam detection at the mail server uses a mix of list-based and content filtering techniques; however, the emails classified incorrectly escape those filters and land in users' inboxes. This can be attributed to the fact that address and keyword lists should account for each individual user's preferences which is not possible at the server level.

If email spam filtering is applied at client side, it is sent to a labelled folder whereas if the filter is applied at the mail server level, messages for different users are either labelled as spam or are deleted providing limited ways in which email spam can be managed (Guzella & Caminhas, 2009).

It is important to consider the concepts of false positives and false negatives again before discussing the problem further. False positive is the legitimate message that is mistakenly identified and marked as spam. False negative is a non-legitimate message not identified as spam but marked as a 'legitimate non-spam message'.
Major problems, for instance, arise when anti-spam techniques misjudge or misclassify legitimate emails as spam (false positive) or fail to deliver or block spam on the SMTP server (false negative), thus causing significant cost in loss of time, effort and monetary terms. False positive cause loss of important information for users while false negatives negate the purpose of the spam filter. False positives and false negatives still pose considerable problems, especially the latter. Misclassifying a legitimate mail as spam has far greater negative impact than letting a spam message pass the filter (L. Yang et al., 2006).

During the process of email classification, there is probability of making such errors, more so at the server level due to its generic nature (Kolcz et al., 2006). Hence, there is need to find a reasonable trade-off between the two types of errors, -having spam emails in the inbox versus losing valuable information. Since most of the business transactions in an organization are conducted through emails, losing an important email can have significant consequences on the core business of an organization and may lead to revenue losses. For example, potential students enquire about courses at a university. If such emails are lost as spam, the university may suffer loss of revenue and reputation. On the other hand, classifying several spam messages as legitimate mail defies the purpose of having a filter. Hence, it is important to have a reasonable trade-off between the two. For instance, if the aim is to reduce the probability of false positives to zero, then is it enough not to filter messages at all. Also, if it is desired that the probability of false negatives be equal to zero, is it sufficient to classify all messages with a spam factor (weak spam)? Obviously, the two extremes are unacceptable.

Spam can be classified as an organizational and an individual problem. It must be acknowledged that server-based filtering is beneficial to address organizational email spam problems and filter the generic spam. Client-based filtering at another level is more suitable to individual user email spam problems. Implementing only client-side filtering solutions may not eliminate all email spam that targets organization. Solution that is closer to the problem is better, hence email spam detection at server side is advisable.

Hence, to eliminate this problem, another layer of filtering at the client level - as a client-based solution – would be advantageous. To further address this issue, a closer look at a filter model is important. Server-based filters use a generic model defined for all whereas client-based filters are personalized for each individual user. Mere reliance on business rules does not suffice.

3.5 Spam filtering and Reactivity of Spammers

The field of software development is continuously evolving, and spam filtering is following the same pattern. Since customer needs are changing due to the evolving nature of businesses, the need for spam filtering also constantly changes due to spammers capacity to react quickly to changes in spam filtering.

Though a significant number of techniques have been suggested for spam filtering (see Chapter 2), content filtering has become more important. The current state-of-the-art in spam filtering consists of a combination of several techniques. The most popular machine learning methods used for content filtering are Bayes classification, Support Vector Machine, k-Nearest Neighbor, Neural Network, Decision trees, Hidden Markov and others. Though researchers have done work on other techniques and have noticeable contributed to the area of email spam filtering, literature suggests that Bayes classification is superior given the speed and accuracy of email spam filtering it provides. It serves as the basis for many commercial and non–commercial solutions for email spam filtering. However, Bayes classifiers are often used in combination with other techniques (i. e., a Bayes classifier is combined with listing techniques).

Over the past decade, spam filtering methods have become increasingly sophisticated; however, so have spamming techniques that reduce the efficiency of filters. Unfortunately, they are highly successful as many false negatives escape filters and reach the inbox of users. Researchers are generally quick to respond to new challenges, aided by technology. As discussed, a remarkable capability of email spam filtering is the learning ability of machine learning algorithms such as Bayes classifier providing a counter measure. However, once a counter measure to a spamming activity has been created, spammers are quick to react with a new way of deceiving filters. Such techniques include tokenization, obfuscation, concept drift, poisoning attacks and many more. Hence, one of the major problems in email spam filtering is the capacity of spammers to react rather than the spamming activity or attack itself.

3.6 Context Specific Training Datasets

Most content filters are trained using datasets that contain a collection of generic spam and diverse ham sample messages which results in a relatively broad feature set for calculating the probabilities of the tokens used for building the classification model. Since a significant amount of email traffic comes from business emails, focus must be on the needs of business email users in an organization. An organization may belong to a particular industry type of which a broad categorization of industry types is finance, banking, healthcare, education, automobiles and so on. Each organization belonging to a particular industry type would have a specific collection of emails that are termed as ham. That means that all ham emails contain the same semantic features that are typically used in that context (Gargiulo et al., 2009). In any context, analysis of the word localization of an entire email at a semantic level effectively measures the weight of a word in a single email. Therefore, a training set that contains ham emails coming from a generic dataset would not apply to a specific context and the classification model performance to correctly classify the new emails would be compromised (K. Junejo & Karim, 2013). Hence, a training dataset containing a collection of emails based on information exchange within that industry type yields better results.

To test these proposed enhancements, in further sections a proof of concept is realized to demonstrate that context specific training datasets improve the performance of email classification. This requires experiments to be conducted to train client-side email filters with context and user specific datasets. The domain is ‘educational’ context. The filter is trained with context specific 8000+ ham and 4000+ spam emails. The outcome of this training was evaluated on new incoming emails, with results showing promise with improvement of the false positive rate by 86% i.e. the emails that were earlier landing in the user’s inbox as spam emails were now detected by the retrained client-side filter with context and user specific dataset. The following section provides details of the proof of concept experiment.

3.7 Experiment 1 - Client-side filtering with Context Specific Training datasets: A Bayesian Classifier

The main motivation for this experiment is the fact that individual user's interests are different. Some well-designed filters (for example, Bayesian Filters) work well, having considerable success under certain conditions; however, this is not static. Spammers are able to vary the success rate with ease, evolving new techniques to deceive filters.

Two factors in particular impact the success rate of spam filters. One of these arises from the fact that a user may belong to a particular industry type - referred to as 'Context". The other element is that users' have individual preferences. For a user in a particular type of organization, the typical collection of emails belonging to that domain will be 'ham' in most cases; however, due to users' individual preferences there are variations. Here, both factors are addressed, and the following hypothesis is put forward:

Context and user specific training datasets improve the classification performance of email spam filters at the client side and reduce false positives.

To prove this hypothesis, an experiment was conducted to verify that the capability of the filtering model can be enhanced with context and user preference specific training datasets. This leads to an increase in the performance of the spam filter on the email client side. For the experiment, a Bayesian classifier was used as an email filtering tool on the email client side of a user, trained with a context-specific dataset and performance monitored over 12 months.

The aim was to ascertain whether such training can reduce the false negatives (email spam) in the inbox of an individual user. Hence, the classification filter was trained with context and user specific datasets and performed classification tasks on incoming new emails for evaluation. Results were then analysed to determine if such training improves the classifier performance and reduces false negatives.

The argument underlying the experiment was built around the fact that there is lack of correlation between the receiving user's area of interest and content of spam email used for training the filters. There are many email spam corpora repositories for training filters with collections of spam and ham emails which. These become the training data sets that can be fed into the filter to create a feature set library while filters at the mail server are trained with generic spam and ham data. Many researchers have studied the content of spam messages and a range of categorisations have been published. One of the categorisations based on content type is Scams, Adult, Financial, Pharmaceuticals, stock, phishing, diploma, software Malware, gambling, dating and others (SpiderLabs, 2013). Email spam detection relies on such categorisations present in datasets to train content filters, whereafter the filters look for features related to any of those categories in any email received. Such training datasets contain features from a mix of these categories and the features may be confusing for a filter. For a new email feature set, the classifier considers everything other than the features found from these categories as ham. In fact, the characteristics and features present in collections of emails for users differ from one context to another. For example, a user that belongs to the healthcare context is likely to receive healthcare related emails as compared to a user who belongs to the real estate industry. Hence, the features learnt by the filtering system are different too.

Training of filters with different feature selection methods is addressed in Gomez and Moens (2010) and training filters at the client side using client data is not totally new; any user who installs an add-on client filter can train it with training data (Meyer & Whateley, 2004). An organisation that uses add-on client-based filters to sieve incoming emails, would have to retrain the filter on all received email for multiple users (Nelson et al., 2009) - an expensive exercise.

Hence, this research aims to identify the percentage impact of context specific training datasets used to train email filters on email spam that an individual user receives in their inbox - at the client-side.

Method: In order to test the hypothesis, this Bayesian filter as client-side email spam filter was installed on a Microsoft outlook email client and trained with a context specific dataset. A Bayesian classifier was chosen due to the use of pure machine learning methods for filtering, good accuracy, validity, popularity and its familiarity with the academic community (Meyer & Whateley, 2004; Nelson et al., 2009).

The novelty in this experiment is that the dataset developed for training the filter was designed to contain emails that incorporate users' email preferences. The dataset contains a collection of spam and ham emails classified by an individual user for this purpose. The filter was trained to avoid correlation between the receivers' domain area and the email content - the email was unwanted by the user. Details of the Bayesian filter and the experiment using specified datasets are discussed below.

3.7.1 The Bayesian Classifier:

This Bayesian classifier is a content-based spam filter that classifies messages based on tokens (including header tokens) observed in an email. It is, therefore, of importance to understand the training model and learning methods of this classifier. The classifier is based on Bayes Theorem (Toit & Kruger, 2012; Liang & Yu, 2012; Vu Duc & Truong Nguyen, 2012) that calculates the probability of occurrence of each word in the document:

Naïve Bayes

The Bayesian classifier is based on the Naïve Bayes algorithm which was first developed by the 18th century mathematician Thomas Bayes (1702-1761) from whom it takes its name. It is a probabilistic machine learning algorithm that performs well on large datasets, is based on Bayes theorem used for predictive classification tasks, and is a simple but powerful model with a naive assumption of statistical independence among features (V. Metsis, 2006). It assumes that there is no relationship among features present in a particular class; thus, no inferences can be drawn from one feature about another. The theorem is based on the conditional probability that if an event can occur than another event has already taken place. The theorem allows the calculation of the probability of an event belonging to a certain class - in the form of a hypothesis, given some prior knowledge and the frequency at which the new event occurs. For email spam filtering, the theorem is used to calculate the probability that a particular email is spam or ham.

The theorem is stated as $P(x|y) = \frac{(P(y|x) * P(x))}{P(y)}$ where x is the hypothesis or class or target, $y = (y_1, y_2, y_3, \ldots y_n)$ constitutes predictors or features or evidence which is prior knowledge or the event that has already occurred; $P(x|y)$ is the probability of hypothesis x given y, called posterior probability; $P(x)$ is the probability of hypothesis x; $P(y|x)$ is the probability of predictor y given hypothesis x, called likelihood and $P(y)$ is the a-priori probability of predictor y.

Naïve Bayes Classifier:

It calculates the probability that a vector of features belong to a particular class (Androutsopoulos, et. al. , 2000b). For vector $y = (y_1, y_2, y_3, \ldots y_n)$, its probability is $P(x_m \mid y_1, y_2, y_3, \ldots y_n)$ where m is the number of classes

From Bayes theorem,

$$P(x_m|y) = \frac{(P(y|x_m) * P(x_m))}{P(y)}$$

In the case of the email spam filtering problem, there are two classes (x): spam, S and legitimate, L and a probability distribution corresponding to each class; $P(y|x)$ represents the probability of am email y with feature vector $(y_1, y_2, y_3, \ldots y_n)$, from class x.

$$P(x|y) = \frac{(P(y|x) * P(x))}{P(y)} = \frac{(P(y|x) * P(x))}{P(y|S) * P(S) + P(y|L) * P(L)}$$

where $P(y)$ is a-priori probability of email y and $P(x)$ is a-priori probability of class x and with known values for $P(x)$ and $P(y|x)$, $P(x|y)$ is calculated for spam emails as $(P(S|y)$ and legitimate emails as $(P(S|y)$ though it is difficult to calculate P(S) and P(L), an approximation only can be developed for each, based on training data as the total number of spam or legitimate email messages verses the total number of email messages. To calculate $P(y|x)$, given a word *w* is present in email message, the feature vector *y* can be defined as 1 if the word is present and 0 if it is absent. So

$$P(y|L) = \frac{\text{frequency of legitimate email with word } w \text{ in it}}{\text{total email messages}}$$

The classification can, thus, be based on probability values of $P(S|y)$ and $P(L|y)$, select the one with highest probability. If $P(S|y)$ is greater, then the email is classified as spam or, otherwise as legitimate; this is the maximum probable hypothesis known as maximum a-posterior probability (MAP) (Tretyakov, 2004).

$$\text{MAP}(S) = \max\big(P(S|y)\big) = \frac{\big(P(y|S) * P(S)\big)}{P(y)} = \big(P(y|S) * P(S)\big)$$

as $P(y)$ being a normalising term can be dropped when the most probable hypothesis is achieved as it is a constant.

From this, a MAP rule is developed that an email is spam if

$$\frac{P(y|S)}{P(y|L)} > \frac{P(L)}{P(S)}$$

else it is a legitimate.

Hence, the principles of probability calculation for a single or multiple features by the Naive Bayesian algorithm (Vu & Nguyen, 2012) are used as follows:

Let the content of each email be called: document.
Class spam email is called 'spam' and
Class ham email is called 'ham'.
The probability that an email is spam is

$$P(spam|document) = \frac{P(spam) * P(document|spam)}{Total}$$

where $Total$ is calculated by

$$P(document|spam) * P(spam) + P(document \mid ham) * P(ham)$$

$$P(document \mid ham) = \prod P(feature_i \mid ham);\ 1 < i < n$$

$$P(document \mid spam) = \prod P(feature_i \mid spam);\ 1 < i < m$$

$$P(spam) = total\ number\ of\ spam\ messages \mid total\ messages$$

$$P(ham) = total\ number\ of\ ham\ messages \mid total\ messages$$

3.7.2 Bayesian Classifier Training Model

The architecture of this classifier acts on tokenisation where an email message is broken into tokens - individual words in the message referred to as features. Tokens are allocated scores and the combined score of the message is calculated. Subsequently, it compares the combined score against a threshold to classify the message.

The occurrences of each token in spam and ham emails are counted to learn which tokens are more indicative of each class. To predict whether a new email is spam or not, a statistical test determines whether the email's tokens are sufficiently indicative of one class or the other and returns its decision or a verdict of 'unsure' is used. The following section details the statistical method the classifier uses to learn token scores before combining them into predictions, but here a realistic model for deployment is discussed.

The training model starts by training a set of labelled messages as spam or ham, which is fed into the filter to generate a classification model. This model (or filter) is subsequently used to classify future email messages. This classification model has multiple output classes: (1) messages that are clearly classified as spam are labelled 'spam' and moved to the spam folder (2) those clearly classified as useful are labelled 'ham' and routed to the Inbox (3) there is also has a third output class - when the tool is not confident it will return an output of 'unsure' which labels the email as spam suspect. The following terminology is adopted: the true class of an email can be ham or spam, and the classifier produces the labels ham, spam, and grey (spam suspects).

There are three natural choices for how to treat unsure-labelled messages: they can be placed in the spam folder, they can be left in the user's inbox, or they can be put into a third folder called Spam Suspects for separate review by the user. The user can then go through this folder of unsure messages to either mark them as spam or ham. Sometimes classifying them as spam or ham can confuse the classifier as those messages contain mixed features of spam and ham. Hence, another choice is to leave them as it is in unsure folder as the purpose here is not to contaminate the training of the filter with these messages.

3.7.3 Bayesian Classifier's Learning and Classification Method

Intuitively, the classifier learns how strongly each token indicates ham or spam by counting the frequency of emails that the token appears in. When classifying a new email, it looks at all of its tokens and uses a statistical test to decide whether they indicate one label or the other with sufficient confidence; if not, it returns unsure.

It considers header and body of an email for classification, tokenizes each email *E* based on words, URL components, header elements, and other character sequences that appear in E. Each of them is treated as a unique token of the email. The algorithm only records whether or not a token occurs in the message, not the frequency of the token. The training and classification method are represented in the Figure 3.7 below:

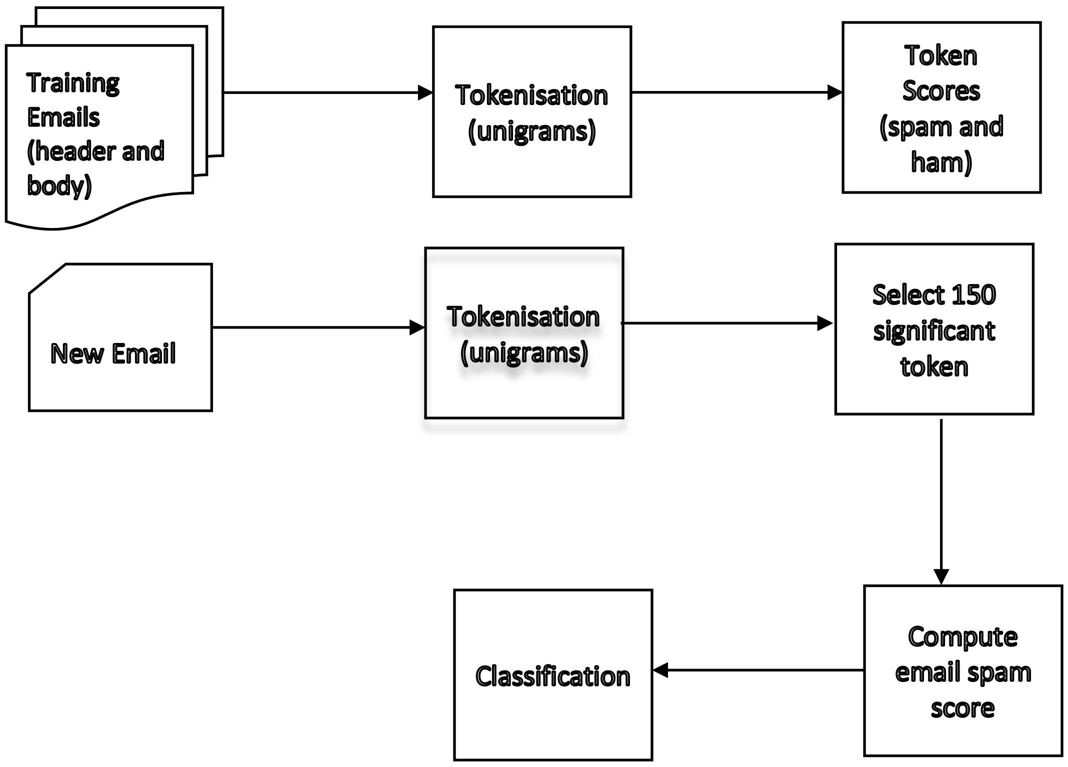


Figure 3.7: Learning and classification Method of Bayesian Classifier

Email *E* is represented as a binary vector *e* where

$$e_i = \begin{cases} 1 & - \; if\ i^{th}\ token\ occurs\ in\ E \\ 0 & - \; otherwise \end{cases}$$

Here *e* is used to refer to the original message E and its representation since the main concern is the latter.

During training, the algorithm computes a spam score vector $P(S)$ where the *i*th component is a token, spam score for the *i*th token is given by

$$P_{(s,i)} = \frac{N_H N_{s(i)}}{N_H N_{s(i)} + N_S N_{H(i)}} \qquad (1)$$

where N_S, N_H, $N_{S(i)}$, and $N_{H(i)}$ are the number of spam emails, ham emails, spam emails including the ith token and ham emails including the ith token, respectively. The quantity $P_{(S,i)}$ is an estimate of $Pr(E\ is\ spam \mid e_i)$ if the prior of ham and spam are equal, but in this case, it is simply a per-token score for the email. An analogous token ham score is given by

$$P_{(H,i)} = 1 - P_{(S,i)}$$

$P_{(S,i)}$ is smoothed through a convex combination with a prior belief x (default value of $x = 0.5$), weighting the quantities by $N_{(i)}$ (the number of training emails with the i^{th} token) and s (chosen for strength of prior with a default of $s = 1$), respectively(Robinson, 2003):

$$f_{(i)} = \frac{sx+N_{(i)}}{s+N_{(i)}} P_{(S,i)} \qquad (2)$$

Effectively, smoothing reduces the impact low token counts have on scores. For rare tokens, the score is dominated by the prior x. However, as more tokens are observed, the smoothed score gradually shifts to the score in Eq. (1). An analogous smoothed ham score is given by $1 - f$.

After training, the filter computes the overall spam score $S(m)$ of a new message M using Fisher's method for combining the scores of the tokens observed in M. The maximum number of tokens chosen to be used here from message M, are 150; the tokens with scores furthest from 0.5 and outside the interval [0.4,0.6] are selected. Let $\delta(m)$ be the binary vector where $\delta(m)i = 1$ if token i is one of these tokens, and 0 otherwise. The token spam scores are combined into a message spam score for M by

$$\mathrm{S(m)} = 1 - \chi^{2}{}_{2n} \left(-2(\log \mathrm{f})^{T} \delta\,(\mathrm{m})\right) \qquad (3)$$

where n is the number of tokens in M and $\chi^2_{2n}(\,\bullet)$ denotes the cumulative- distribution function of the chi-square distribution with $2n$ degrees of freedom.

A ham score $H(e)$ is similarly defined by replacing f with $1 - f$ in Eq. (3). Finally, an overall spam score for M is constructed by averaging $S(m)$ and $1 - H(m)$ (both being indicators of whether m is spam) giving the final score.

$$I(m) = \frac{1+\mathrm{S(m)} - \mathrm{H(m)}}{2} \qquad (4)$$

for a message; a quantity between 0 (ham) and 1 (spam). This Bayesian classifier finally makes prediction by thresholding $I(m)$ against two user-tuneable thresholds θ_0 and θ_1, with defaults $\theta_0 = 0.15$ and $\theta_1 = 0.9$. The classifier predicts ham, grey, or spam if $I(m)$ fall into the interval $[0, \theta_0]$, $[\theta_0, \theta_1]$, or $[\theta_1,1]$, respectively, and filters the message accordingly.

3.7.4 Data, Method and Results

This experiment was conducted with the aim of testing if the training of this Bayesian classifier tool with context specific data and user preferences is effective for the selected domain. Individual user data was collected for this experiment in this domain with extra care taken to segregate spam and ham.

During the experiment, the server-side email filtering was also in place which filters incoming emails for multiple users of the organisation, but false negatives were expected to escape the server and reach the mailbox of the email client. The reason for this is that the features identified by filters are common to all users and secondly, that those filters have not been trained on the basis of context specific keywords/features. In this experiment, it was tested if this Bayesian classifier tool can provide client-based filtering in addition to the existing content filtering at the mail server. This means that the Bayesian classifier filters false negatives that have escaped the content filters at the mail server. This section is structured as Dataset, Method and Outcome.

3.7.4.1 Training Dataset

In an attempt to catch the spam messages coming to the inboxes, the use of appropriate samples to train the filter is vital. The following questions are also of high importance: is the available body of data sufficiently large to reach a conclusion? Is the dataset too broad? Will it capture and feed the intended behaviour for which it needs to be trained into the filter? Is sample data unbiased?

Since the focus is on a user belonging to an organisation in a particular industry domain, it is useful to refer to the worldwide business emails count in reference to the size and nature of the training dataset. Given the total amount of business emails sent per day, (Levenstein, 2013) (given in Chapter 2), if a training corpus consists of 100,000 messages, this still only constitutes a very small portion (approximately 1%) of business email traffic globally. Hence, the dataset used to train the filter represents an extrapolation of generalisations made in the training dataset (Pitsillidis, Kanich, Voelker, Levchenko, a& Savage, 2012).

In this case, to prepare a dataset that is not too large or too small, the issue of biased data is ignored since the data is specific to a domain context. The focus here is to train the filter so that it is capable of identifying incoming emails belonging to this domain as ham.

Based on these considerations, the false negatives (email spam) that arrived in the inbox of a user were collected over 5 years in order to prepare a dataset for this research. A separate folder called 'InboxJunk-Spam' was created and once determination about an email being spam was made; the message was moved to that folder. Since the domain under consideration was 'Educational', which is a broad domain, careful attention was paid while classifying the unwanted messages arriving in the inbox so that the filter was not contaminated with those messages that belong to the same domain but represented spam for this user. Hence, email messages that belonged to the education domain but were unwanted by the user were classified as spam and also moved to folder InboxJunk-Spam. Hence, the user did not continue to receive spam emails though they were related to the domain. The final collection count of messages was 13146 with 8723 ham email messages and 4423 spam email messages, to be used for training the Bayesian classifier.

3.7.4.2 Experimental Method

For experimental purpose, the Bayesian classifier was installed on the Outlook client of a user. The next step was to train the classifier. The training of the tool was carried out with the collected context specific user preference dataset of 8723 ham and 4423 spam messages.

Once the training was complete, the classifier performed classification to filter an incoming email message on the basis of this training. For each incoming email, the Bayesian classification tool first performed tokenisation of the header and body of the message followed by feature extraction of 150 significant tokens, and then computed the spam score of the message. It then classified and categorised the message into one of the three output classes - ham, spam or grey according to the threshold values of 0.15 and 0.9. If the spam score was more than 0.9, it was classified as spam, less than 0.15 it was classified as ham and if it lay between 0.15 and 0.9, it was classified as grey. If misclassified, they end up in one of the three, 1) user's inbox as spam (false negative), 2) spam folder or 3) grey folder as false positive. To observe and record the results correctly, several folders were created and managed. In order to evaluate the performance of the filter, the email spam that was not detected by the Bayesian classifier was kept separately in a folder so that future messages arriving in the inbox as false negatives would not get mixed with those correctly classified. The folders to observe the results of training and evaluation were as follows:

Bayesian Classifier Filtered Spam – This folder collected all emails for this user that had been classified as spam, had passed through the organisational filter and been received by the email client of the user. These emails were the ones that were not detected as spam by the mail server filters. These were incoming emails aiming to land in the user inbox but were detected by the classifier as spam and moved into this folder.

InboxSpam- This folder collected the emails that were false negatives due to not being detected by the organisational mail server filters, stayed false negatives as not detected by the classifier and landed in user's inbox as legitimate emails. The trained tool could not identify these emails as spam. Such emails were manually moved from the inbox to this folder.

Spam Suspects (Grey) – This folder was created to collect emails that were suspected by the classification tool to be spam. A user has the option to manually classify email messages in this folder and report the emails to the classifier as spam or ham for learning. This folder might also contain false positives.

The information from these folders was then used by the Bayesian classifier to further train the filter.

The performance of the classifier was observed for a period of 12 months to identify if there was an improvement in the number of email spam arriving at the user's inbox.

At the end of the experiment, the classification filter was retrained with the updated collection of 11529 ham and 5042 spam messages. Hence, the classifier was retrained with additional ham and spam messages collected in this12 months period.

3.7.4.3 Results

The following results were recorded.

The folder 'Bayesian Classifier filtered Spam' collected 683 emails that were going to the inbox as false negatives. The folder InboxSpam collected 170 emails which were not identified by the classification tool as spam and were false negatives.

The folder Spam Suspects collected 408 emails that were otherwise going to the inbox as false negative. Out of these spam suspects, the user identified 10 emails as false positives, unimportant emails that would, if missed, not impact the productivity of the user. An important point to note is that once the classifier was trained with the initial dataset, it was deliberately not retrained until the end of the observation period to not contaminate the learning of the filter with further data.

The above experiment proved the hypothesis as follows: Training the Bayesian classifier tool on a context specific user preference dataset, the instances of false negatives in the user inbox were reduced by 86%, representing savings in time and increases in user productivity and motivation (K. Bajaj & Pieprzyk, 2014).

3.7.4.4 Discussion:

Among the recommendations to be made as a result of this experiment is to re-emphasise the need to train the filters with user preferences and context specific data so that unwanted emails that pass through the filters can be isolated. This makes the filters at the client level more effective. Although this may not stop the false negatives completely, as noted in this experiment, as 14% of email spam still passed through despite specific training of the Bayesian classification filter. However, it did deter a large number of email spam from reaching the user and reduced the inconvenience caused by spamming. Analysing the 14% email spam that deceived the specifically trained client-side filter, it was revealed that most of those emails belonged to the 'education' domain only. Hence, context specific training was effective, and it can be deduced that training the tool with context specific and user preference data did eliminate most false negatives from the inbox that were unrelated to the domain such as emails containing content selling viagr@ or other pharmaceutical items, banks, holiday deals etc. which were detected by the trained tool. This was a first step towards a client-side classification filter model; however, there is room for improvement and further research and experiments were carried out and are shown in following chapters.

3.8 Experiment 2: Performance Testing of Client-side Bayesian Classifier with n-gram Parametrisation

Naive Bayesian classifier identifies tokens as maximum discriminators. The sequence of any n characters, tokens or words that appear in a message, is represented as an n-gram. A window, as wide as an n-gram, slides over an entire email message to calculate the total number of occurrences of each n-gram in each email. This data is then used as tokens to calculate the spam score of an email. Unigrams are one-word representations. During feature selection, our Bayesian classifier uses unigrams from the header and body of an email. These tokens are selected from training and testing data based on the probability of occurrence of a particular unigram. To classify an email, this classifier selects only 150 significant unigram (single word) tokens. Naïve Bayesian classification assumes that tokens in a given class are independent of each other; in this section experiments are conducted to identify the best performing parameters of this Bayesian Classifier. To do so, several parameters were fine-tuned to analyse and monitor the performance against numerous threshold values, token types and token sizes.

Experiment details:

The experimental technique includes modification of the features used for classification by tweaking 1) n-grams used as text features to include two and three words as tokens called bigrams and trigrams respectively 2) number of tokens selected for classification of a new message, these tokens are termed here as max discriminators.

Though Naïve Bayesian classification considers tokens as independent from each other, correlations are possible between different tokens of an email. To identify these correlations, a correlation matrix was created in these experiments for bigrams and trigrams. However, if a correlation matrix for every word with every other word in a message were created for bigrams and trigrams, the size of the matrix would consume too much memory which would slow down the classifier performance. It was, therefore, decided to consider only the correlations between neighbouring tokens for both, bigrams and trigrams.

For example, if a, b and c are the three tokens then bigrams are ab, bc, a, b, and c. The frequency of bigrams is calculated as the number of times they appeared in spam and ham emails. Trigrams were created in a similar manner, if a, b, c, d and e are the three tokens then trigrams are abc, bcd, cde, ab, bc, cd, de, a, b, c, d and e. Subsequently, all tokens (unigrams, bigrams or trigrams) were sorted according to their importance and only the first n significant tokens were taken to calculate the score of an email.

The Bayesian classifier uses the default threshold of 0.15 and 0.9 for classification. Several threshold values for spam and ham cut off were used to find the best performing cut offs. Threshold value sets used are shown in Table 3.1.

Table 3.1. Threshold Values for Spam and Ham cut off

Ham Cut-off	0.5	0.15	0.2	0.3	0.8
Spam Cut-off	0.5	0.9	.9	0.8	0.6

The dataset in machine learning is the collection of data used to train the machine learning model to automatically perform actions for decision making and predictions. Several datasets are generally used to measure the effectiveness of each of the machine learning models. For the email spam filtering problem, the supervised machine learning model relies on a dataset that contains the collection of labelled spam and legitimate emails to create the classifier.

In order to validate the performance results and to find the optimal values, several types of data input were used in the experiments consisting of four publicly available spam email datasets, PU1 (J. K. Androutsopoulos, Chandrinos, K.V. Paliouras,G & Spyropoulos, C.D. , 2000b), Ling-spam (J. K. Androutsopoulos, Chandrinos, K.V. Paliouras,G & Spyropoulos, C.D. , 2000a), ENRON (V. Metsis, 2006), and TREC07, a dataset published at the Text REtrieval conference 2007 (TREC2007[2]), by the University of Waterloo, Canada.

PU1:This corpus was first released in 2000 and then updated and described in I. Androutsopoulos, Paliouras, and Michelakis (2006) is only in its bare form here, where tokens are separated by white spaces but lemmatisation of stop words has been applied. The corpus consists of 10 partitions and an unused partition. Each of the 10 partitions contain 619 spam and 482 legitimate messages.

Ling-spam: This linguistic mailing list public spam corpus was introduced in 2000 (J. K. Androutsopoulos, Chandrinos, K.V. Paliouras,G & Spyropoulos, C.D. , 2000a) and is used for evaluating the performance of the Naïve Bayes machine learning algorithm as a classifier. The corpus consists of a total of 2893 email messages, of which 2412 are legitimate and 481 are spam messages. This dataset is somewhat imbalanced in the sense that it is quite small and secondly it only has 16% spam messages. However, the messages in this dataset are more homogenous when compared to messages found in a usual inbox hence the dataset can be effective for its purpose.

TREC07: This publicly available email spam dataset contains 75,419 messages out of which 25,220 are legitimate emails and 50199 are spam messages (Cormack G V., 2007).

Enron: Enron corpus was made available in 2004 by the Federal Energy Regulatory Commission during their investigation of the Enron corporation. The dataset contains data from 158 users of Enron organised into folders and constituting the only publicly available real email spam dataset (J. K. Androutsopoulos, Chandrinos, K.V. Paliouras,G & Spyropoulos, C.D. , 2000a). Though the entire dataset is large with 619448 messages (Klimt & Yang, 2004), not all are labelled and some

[2] http://plg.uwaterloo.ca/~gvcormac/treccorpus07/

also contain redundant data. The labelled dataset size is 33k belonging to 6 users. The latest update on the dataset was done on 7 May 2015.

Percentage splits for training and testing in existing studies vary; however, this thesis uses a 70-30 split to divide the datasets for training and testing purposes. The method of 70-30 split is called hold out method. The issue with the holdout method is that there is no control over which emails fall into which category which may lead to an imbalance in the datasets and performance evaluation may be affected. Cross validation is an effective way to reduce such shortcomings using k fold cross validation where the entire dataset is divided into k parts. With a 70-30 split, the model is trained with 70% of the data and tested on 30%, performed k times with for each operation.

The totals emails used for each of the datasets is shown in Table 3.2.

Table 3.2: Total Number of Training and Testing Emails for each Dataset used

Dataset Name	Total Emails	Number - Training	Number - Testing
PU1	1,100	770	330
Ling-spam	2,893	2,025	868
Enron	11,935	8,354	3,581
TREC07	20,026	14,862	5,164

There were 85 parameter sets in total for each dataset - 25 for unigrams and 30 each for bigrams and trigrams. For each parameter set, 20 iterations were run, randomly selecting emails for training and testing for the purpose of cross validation. For example, for unigrams token size 15 and thresholds of 0.5-0.5 as spam and ham cut off, 20 iterations for each dataset were executed. Email messages were selected randomly to be allocated to the pool of training and testing sets for each iteration. Similarly, for each token size corresponding to every n-gram as shown in Table 3.3, selected threshold set iterations were run.

Table 3.3 Token Sizes (Max Discriminators) used for 3 n-gram types

Unigram	Bigrams	Trigrams
15	150	150
50	500	500
75	5000	5000
150	10000	10000
200	20000	20000
	25000	25000
		50000

For execution of each dataset, results were recorded for all 20 iterations for all parameter sets. The performance metrics used were False Positive (FP) rate, False Negative (FN) rate and Grey rates. The false positive and false negative rates are defined as:

The False positive rate (FPR) is the measure of accuracy of a classifier and is defined as the probability of falsely rejecting the null hypothesis. In this case, it is the number of incorrect positive predictions (misclassified ham messages) versus the total negatives (ham messages). is calculated as:

$$FP\,rate = FP\,/\,(FP + TN)$$

The False positive rate (FPR) is also called miss rate of a classifier and is defined as the probability of falsely rejecting the null hypothesis. In this case, it is the number of incorrect negative predictions (misclassified spam messages) versus the total positives (spam messages). It is calculated as:

$$FN\,rate = FN\,/\,(TP + FN)$$

The Grey rate is the measure of non-classification of a classifier and is defined as the probability where the classifier is not able to make a decision about the null hypothesis. In this case, it is the rate of unclassified or no predictions made in relation to the email messages.

$$Grey\,rate = \text{unclassified emails}\,/\,(TP + TN)$$

A cost function as a function of FP rate, FN rate and grey rate was calculated to determine the best performing parameters as below:

Cost function = C(*f*) = 0.5 * FP rate + 0.2 *FN rate + 0.3 * grey rate

The cost function is based upon the impact of FP rate, FN rate and grey rate on a user. The weights of 50%, 20% and 30% were attached to FP rate, FN rate and grey rate. These weights were selected since losing an important email as false positive can cause significant damage to a user and hence FP rate was given 50% importance. Grey email may contain a mix of ham and spam whereby losing an important email as spam via this output class and removing email message from inbox, thus given 30% importance. FN rate determines the capability of the filter to deter email spam message from the inbox therefore was allocated a weightage of 20%.

Experiment 2.1: Unigrams with Varying Feature set (token) size

In order to improve the performance of the Bayesian classifier, the first experiment was aimed at working with unigrams to identify the optimum number of significant tokens selected for performing the classification.

Method: In this experiment, unigrams with varying significant token sizes of 15, 50, 75, 150 and 200 for each of the threshold value set given in Table 3.1 was selected for each dataset. The total number of parameters tested for unigrams for each dataset - PU1, Ling-spam, Enron and TREC07 is shown in Table 3.4.

Table 3.4: Parameter Sets for each dataset for Unigrams

Significant token size	Spam cut off	Ham cut off
15	0.5	0.5
15	0.15	0.9
15	0.2	0.9
15	0.3	0.8
15	0.4	0.6
50	0.5	0.5
50	0.15	0.9
50	0.2	0.9
50	0.3	0.8
50	0.4	0.6
75	0.5	0.5
75	0.15	0.9
75	0.2	0.9
75	0.3	0.8
75	0.4	0.6
150	0.5	0.5
150	0.15	0.9
150	0.2	0.9
150	0.3	0.8
150	0.4	0.6
200	0.5	0.5
200	0.15	0.9
200	0.2	0.9
200	0.3	0.8
200	0.4	0.6

10-fold cross validation was performed as shown in Table 3.5 for Ling-spam dataset under column headings ‘Train Folders’ and ‘Test Folders’

Table 3.5: Test Execution results for Ling-spam dataset for threshold value of 0.5 each for spam and ham cut off.

NHAM TESTED	NHAM RIGHT	NHAM WRONG	FALSE POSITIVE RATE [%]	NSPAM TESTED	NSPAM RIGHT	NSPAM WRONG	FALSE NEGATIVE RATE [%	TRAIN FOLDERS	TEST FOLDERS	NUM TRAIN MESS	NUM TEST MESS	TRAIN MESS	HAM CUTOFF	SPAM CUTOFF
724	724	0	0	144	140	4	2.777777778	part3 part8 part9 part7 part6 part10 pa	part1 part2 part5	2025	868	69.99654338	0.5	0.5
724	724	0	0	145	137	8	5.517241379	part2 part7 part9 part8 part1 part5 par	part10 part4 part6	2024	869	69.96197719	0.5	0.5
723	721	2	0.276625173	144	142	2	1.388888889	part2 part3 part9 part1 part10 part6 pa	part8 part4 part7	2026	867	70.03110957	0.5	0.5
723	723	0	0	144	137	7	4.861111111	part5 part10 part2 part7 part4 part8 pa	part1 part6 part9	2026	867	70.03110957	0.5	0.5
724	724	0	0	144	142	2	1.388888889	part8 part4 part1 part3 part9 part2 par	part5 part6 part7	2025	868	69.99654338	0.5	0.5
724	723	1	0.138121547	145	138	7	4.827586207	part9 part3 part1 part8 part2 part6 par	part10 part4 part7	2024	869	69.96197719	0.5	0.5
723	722	1	0.138312586	144	142	2	1.388888889	part2 part6 part10 part7 part4 part9 pa	part8 part1 part3	2026	867	70.03110957	0.5	0.5
724	724	0	0	144	143	1	0.694444444	part6 part10 part2 part1 part8 part9 pa	part3 part5 part7	2025	868	69.99654338	0.5	0.5
723	723	0	0	144	142	2	1.388888889	part10 part5 part9 part8 part4 part2 pa	part1 part3 part7	2026	867	70.03110957	0.5	0.5
724	721	3	0.414364641	144	142	2	1.388888889	part7 part9 part4 part10 part3 part6 pa	part8 part2 part5	2025	868	69.99654338	0.5	0.5
724	723	1	0.138121547	145	137	8	5.517241379	part4 part8 part1 part2 part3 part7 par	part9 part10 part6	2024	869	69.96197719	0.5	0.5
724	724	0	0	144	143	1	0.694444444	part6 part1 part2 part9 part8 part10 pa	part4 part5 part7	2025	868	69.99654338	0.5	0.5
723	723	0	0	144	137	7	4.861111111	part7 part8 part10 part3 part6 part4 pa	part1 part2 part9	2026	867	70.03110957	0.5	0.5
724	722	2	0.276243094	145	140	5	3.448275862	part6 part9 part1 part2 part7 part4 par	part8 part10 part3	2024	869	69.96197719	0.5	0.5
723	721	2	0.276625173	144	141	3	2.083333333	part5 part3 part4 part7 part10 part9 pa	part8 part1 part6	2026	867	70.03110957	0.5	0.5
724	724	0	0	144	142	2	1.388888889	part8 part1 part10 part3 part2 part4 pa	part5 part6 part7	2025	868	69.99654338	0.5	0.5
724	722	2	0.276243094	145	138	7	4.827586207	part4 part8 part2 part1 part6 part7 par	part9 part10 part3	2024	869	69.96197719	0.5	0.5
723	722	1	0.138312586	144	139	5	3.472222222	part9 part1 part5 part3 part8 part6 par	part2 part4 part7	2026	867	70.03110957	0.5	0.5
725	725	0	0	145	139	6	4.137931034	part6 part3 part4 part8 part2 part9 par	part1 part10 part5	2023	870	69.92741099	0.5	0.5
723	723	0	0	144	137	7	4.861111111	part5 part4 part10 part3 part6 part7 pa	part1 part2 part9	2026	867	70.03110957	0.5	0.5

The Bayesian classifier was trained using 70% of the email messages from each dataset and the learning of the filter model was evaluated by executing the model for all 25 parameter sets on 30% of the email messages from each dataset and results recorded.

Results:

For unigrams with 15 token size, Bayesian classier performed well for FP rates for all datasets for all thresholds except PU1; however, the difference in values for rates was quite low as shown in Figure 3.8. The grey rate was high, ranging between 6-9.6% for the thresholds that are far apart from the median threshold values for all three datasets except Enron where it was below 1.5%. The FN rate was higher for the 0.5 threshold value compared to other threshold values for spam and ham cut offs for the Ling-spam dataset though it was showing good performance for other datasets.

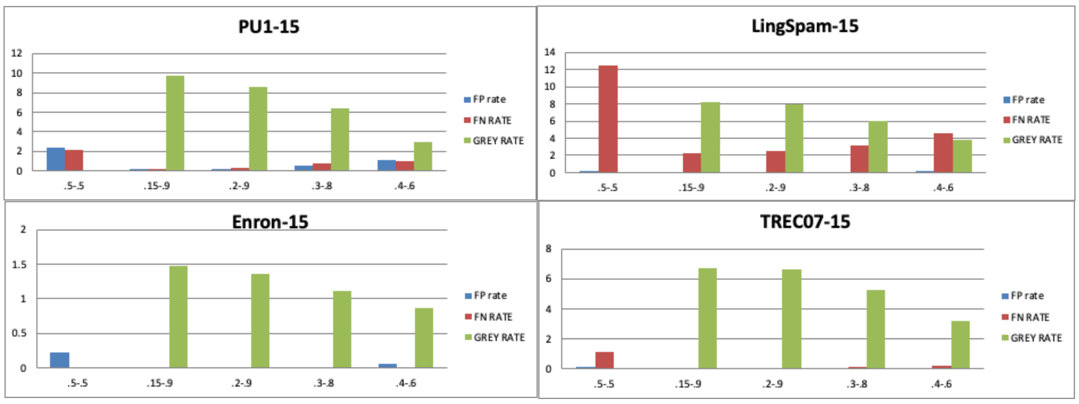


Figure 3.8: Test Results for all four datasets for Unigrams with token size of 15 for several threshold sets

The performance of the classifier for unigrams with a token size of 50 was good for the FP rate which was below 0.4% for all three datasets except the PU1 with FP rates between 2-3% as shown in Figure 3.9. The FN rate is on the higher side for the Ling-spam dataset, although for remainder of the datasets the classifier performed well. They grey rate was higher by between 2-8.5% for most datasets.

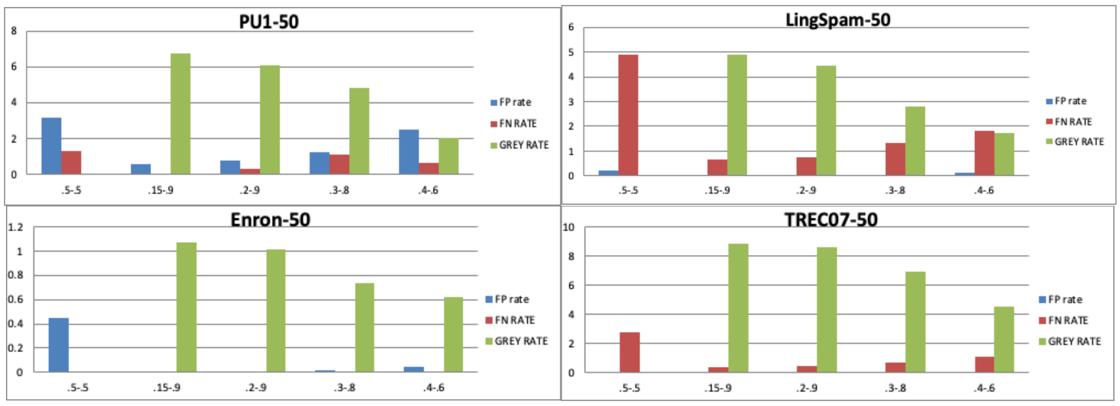


Figure 3.9: Test Results for all four datasets for Unigrams with token size of 50 for several threshold sets

The performance of the classifier for the token size of 75 was similar to the token size of 50 for unigrams as is evident from Figure 3.10 though the FN rate was somewhat higher and the grey rate lower across all threshold values for all datasets.

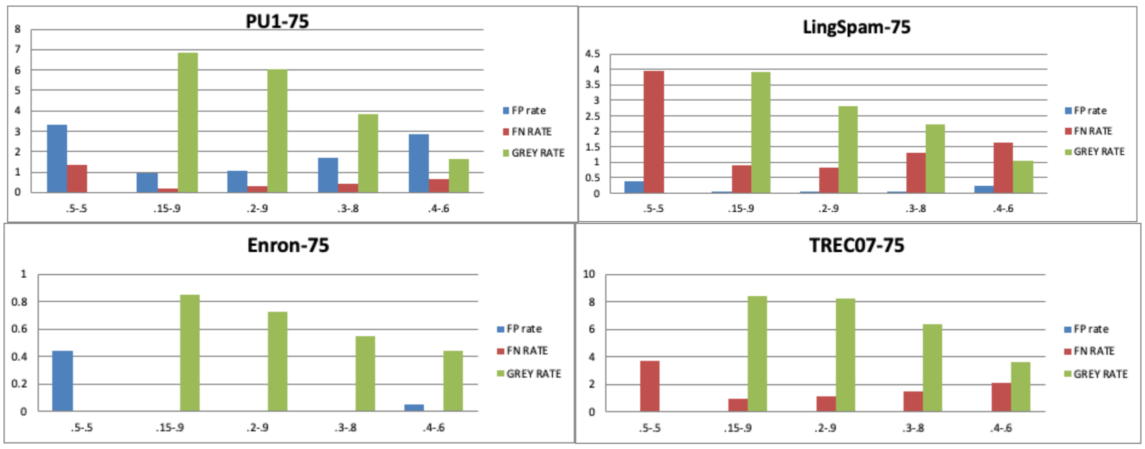


Figure 3.10: Test Results for all four datasets for Unigrams with token size of 75 for several threshold sets

Figure 3.11 represents the classifier performance for token size 150 where the FP rate ranges between 1-4% for PU1 and Ling-spam datasets; however, it is relatively low for Enron and TREC07 datasets. For the FN rate, the classifier performed exceptionally well for the Enron dataset, and well for PU1 and Ling-spam datasets with <1% whereas for the TREC07 the rate was between 2-5% for several threshold values.

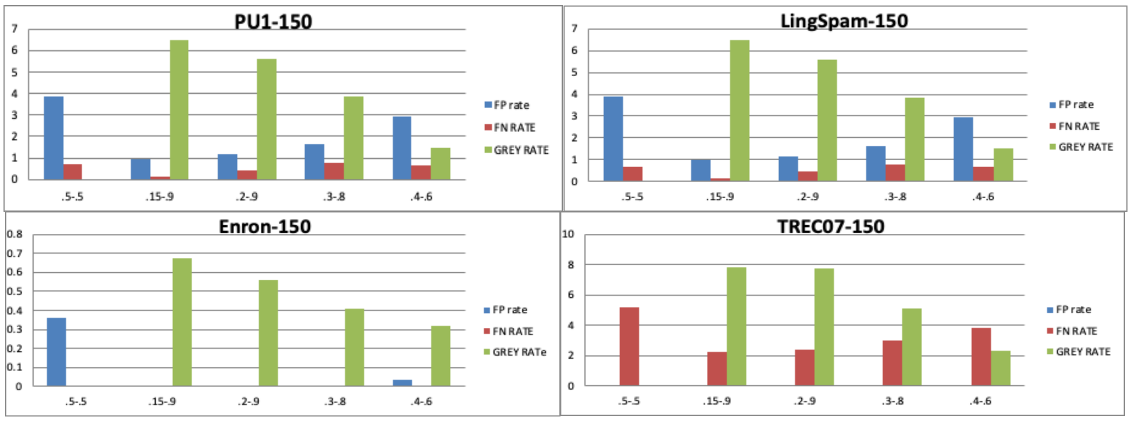


Figure 3.11: Test Results for all four datasets for Unigrams with token size of 150 for several threshold sets

Increasing the token size to 200, the classifier performance as shown in Figure 3.12 improved for the FP rate compared to the token size of 150 however performance for the FN and grey rate declined for all four datasets.

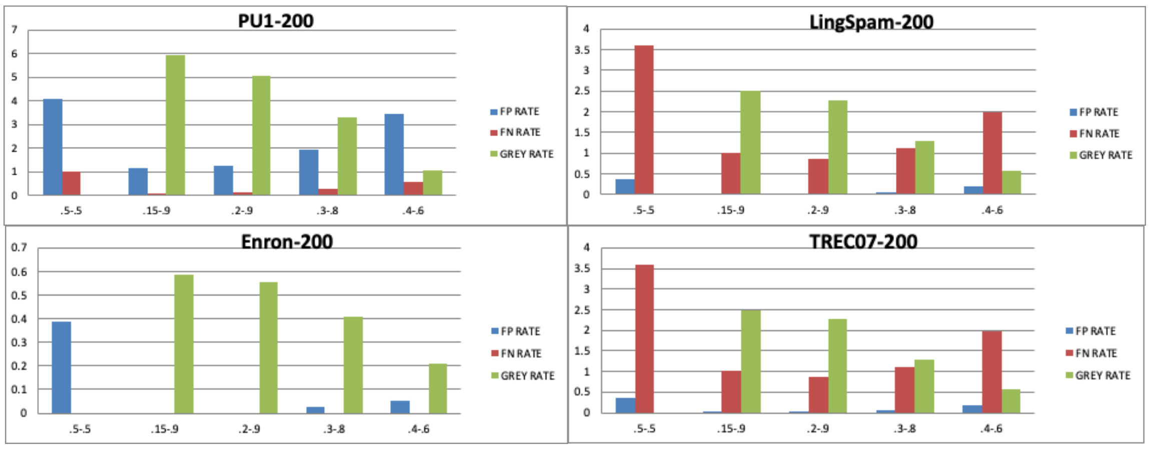


Figure 3.12: Test Results for all four datasets for Unigrams with token size of 200 for several threshold sets

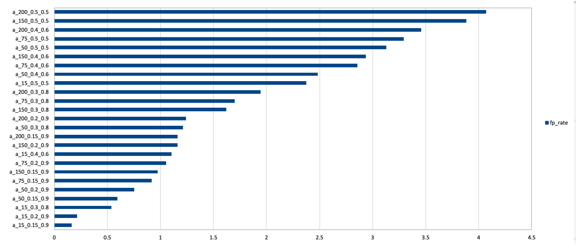
The overall results for all token sizes for all thresholds for each of the four datasets is shown in Table 3.6 for unigrams. The value of the cost function for each parameter set is shown for each of the datasets. Results indicate that the classifier for token size 15 for the Enron and TREC07 datasets performed better whereas classifier performance is comparable for token sizes 150 and 200 for all datasets. For the FP rate, the classifier performance was superior for token size 15; however, the values were very close for all datasets and token sizes for each of the datasets individually. Considering the cost and other metrics the performance is noted to be better for the token size of 150 overall.

Table 3.6: Results for Unigrams for all four datasets for all the parameters

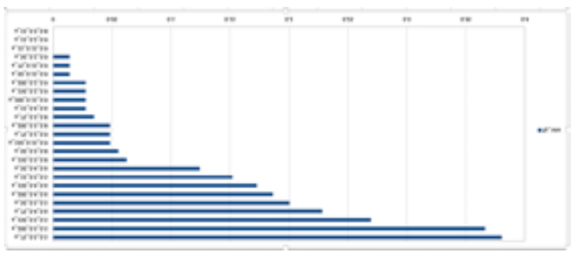
MAX TOKENS	HAM CUT OFF	SPAM CUT OFF	FP RATE PU1	FN RATE PU1	GREY RATE PU1	COST	FP LINGSPAM	FN LINGSPAM	GREY LINGSPAM	COST	FP ENRON	FN ENRON	GREY ENRON	COST	FP TREC07	FN TREC07	GREY TREC07	COST
15	0.5	0.5	2.376654	2.11614	0	1.611585	0.15205788	12.4909004	0	3.808123	0.232358	0	0	0.092973	0.16345	1.103838	0	0.396561
15	0.15	0.9	0.161581	0.173372	9.727609	2.061017	0	2.28927203	8.241518	2.335115	0	0	1.478637	0.295757	0	0.072938	6.732618	1.368435
15	0.2	0.9	0.21549	0.347222	8.554811	1.888182	0	2.52681992	7.908288	2.339734	0	0	1.35437	0.270904	0	0.102001	6.59971	1.350572
15	0.3	0.8	0.539234	0.72773	6.381959	1.691585	0	3.2217433	6.015644	2.169682	0	0	1.118403	0.223711	0	0.139056	5.29134	1.100015
15	0.4	0.6	1.10608	1.005508	2.94401	1.342974	0.02766252	4.53448276	3.7672	2.12488	0.060241	0	0.869869	0.1981	0	0.211331	3.220422	0.707514
			0.879808	0.873994	5.521678	1.719038	0.03594408	5.01264368	5.18653	2.555477	0.05852	0	0.964256	0.216259	0.03269	0.325833	4.368818	0.984589
50	0.5	0.5	3.12949	1.284004	0	1.821646	0.20038142	4.87452107	0	1.542609	0.447504	0	0	0.179102	0	2.759902	0	0.828071
50	0.15	0.9	0.59256	0.034722	6.747146	1.652754	0.01382171	0.6585249	4.880545	1.179295	0	0	1.069534	0.214007	0	0.372941	8.878492	1.887681
50	0.2	0.9	0.754432	0.311542	6.092148	1.658054	0.01381218	0.76077586	4.441126	1.122083	0	0	1.016476	0.203395	0	0.446919	8.574017	1.848979
50	0.3	0.8	1.212002	1.109195	4.819628	1.791866	0.05529638	1.3493774	2.788539	0.98474	0.017212	0	0.740017	0.154988	0	0.720389	6.914191	1.599055
50	0.4	0.6	2.483152	0.659722	2.018791	1.777379	0.12440494	1.83500958	1.750916	0.950548	0.043029	0	0.621335	0.141579	0	1.115941	4.507536	1.23639
			1.634327	0.679837	3.935543	1.74024	0.08154333	1.89564176	2.772225	1.155755	0.101549	0	0.689472	0.178514	0	1.083218	5.774847	1.479935
75	0.5	0.5	3.291082	1.35273	0	1.916237	0.38020681	3.9454023	0	1.335853	0.438898	0	0	0.175709	0	3.741255	0	1.122527
75	0.15	0.9	0.916888	0.208333	6.870512	1.874363	0.01382171	0.90038314	3.900584	1.05591	0	0	0.848925	0.169935	0	0.94406	8.40207	1.963782
75	0.2	0.9	1.051003	0.3125	6.018576	1.791866	0.04837122	0.83141763	2.805536	0.830031	0	0	0.726054	0.145361	0	1.135867	8.227305	1.986371
75	0.3	0.8	1.698925	0.416667	3.849744	1.702895	0.03455904	1.31704981	2.212541	0.851597	0	0	0.548729	0.109896	0	1.449827	6.413275	1.717753
75	0.4	0.6	2.856736	0.659722	1.63853	1.888168	0.22808209	1.62835249	1.036921	0.787273	0.051635	0	0.438425	0.108489	0	2.117802	3.650738	1.365638
			1.962927	0.58999	3.675473	1.834556	0.14100818	1.72452107	1.991116	0.971983	0.098107	0	0.512427	0.141728	0	1.877762	5.338678	1.631064
150	0.5	0.5	3.882605	0.693487	0	2.0803	0.26946125	2.70043103	0	0.918214	0.361446	0	0	0.144878	0	5.212777	0	1.564133
150	0.15	0.9	0.972698	0.13841	6.460026	1.806336	0.04839033	1.00502874	2.869577	0.89508	0	0	0.674393	0.135179	0	2.235901	7.849024	2.240875
150	0.2	0.9	1.160569	0.450431	5.583698	1.78741	0.02764341	0.90158046	2.449261	0.771684	0	0	0.561296	0.112559	0	2.381849	7.719565	2.258767
150	0.3	0.8	1.619014	0.796456	3.85209	1.739516	0.06219293	1.24497126	1.479946	0.694658	0	0	0.411896	0.082679	0	3.01188	5.148665	1.933597
150	0.4	0.6	2.936646	0.658525	1.500375	1.900403	0.17274745	1.69731801	0.633568	0.705308	0.034423	0	0.318347	0.077739	0	3.849089	2.301927	1.615412
			2.114306	0.547462	3.479238	1.862493	0.11608707	1.5098659	1.48647	0.796689	0.079174	0	0.393186	0.110307	0	3.338299	4.603836	1.922257
200	0.5	0.5	4.070339	1.005029	0	2.236575	0.36630866	3.60632184	0	1.22882	0.387263	0	0	0.155305	0	5.174904	0	1.552871
200	0.15	0.9	1.160573	0.103927	5.931314	1.787735	0.02765297	1.00454981	2.494302	0.811687	0	0	0.586428	0.117686	0	2.602435	7.453667	2.271864
200	0.2	0.9	1.239756	0.138889	5.047543	1.657564	0.02763389	0.86494253	2.269496	0.724836	0	0	0.552918	0.110984	0	2.818818	7.315858	2.309217
200	0.3	0.8	1.943056	0.277538	3.322383	1.691912	0.0483712	1.10871648	1.302555	0.612874	0.025818	0	0.409104	0.092548	0	3.368697	4.969044	2.004818
200	0.4	0.6	3.454529	0.589799	1.047115	2.055047	0.18660737	1.97246169	0.576004	0.781982	0.051635	0	0.209439	0.062942	0	3.979116	2.2369	1.641515
			2.373651	0.423036	3.069671	1.885367	0.13131482	1.71139847	1.328472	0.83164	0.092943	0	0.351578	0.107493	0	3.588794	4.395094	1.955657

In order to carry out further analysis of the classifier performance for each parameter, the FP rate and the cost for all token sizes for all thresholds for each of the dataset is represented in graphical form in Figure 3.13.

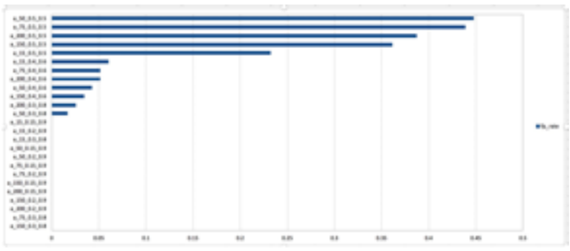
False Positive Rate

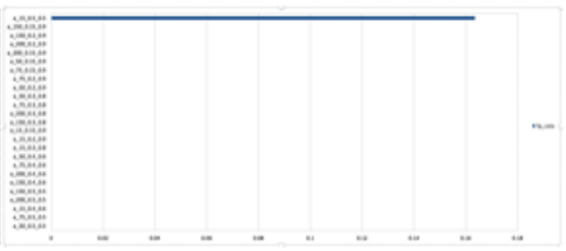


PU1



Ling-spam

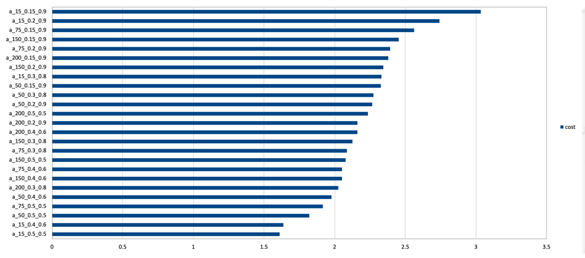


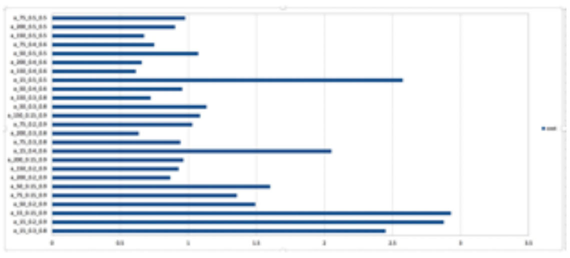


Enron

TREC07

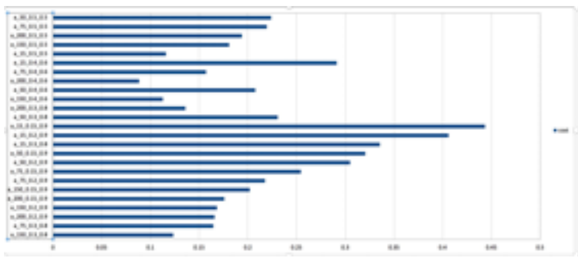
Cost

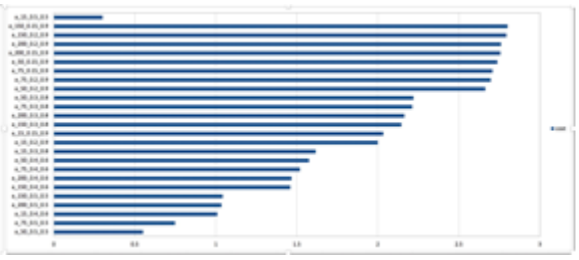




PU1

Ling-spam





Enron

TREC07

Figure 3.13: FP rate and cost function for all four datasets for Unigrams for all parameter values

For far apart threshold values the cost was high and the FP rate low due to the high number of greys for far apart threshold values.

Experiment 2.2 Bigrams with Varying Feature Set (Token) Size

For the second experiment, it was decided to use bigrams as features to determine if this increases the performance of the Bayesian classifier. A bigram is an n-gram where n=2 (i.e. two words) are considered as significant tokens that offer the best cost value for bigrams to compare against other n-grams.

Method: In this experiment, since bigrams can generate a higher number of features for a message, it was decided to have the least number of tokens selected as features for classification as 150 and maximum number of tokens as 25,000. Therefore, significant token sizes of 150, 500, 5000, 10000, 20000, and 25000 for each of the threshold value set given in Table 4.1 were selected for classification and evaluated using all four datasets- PU1, Ling-spam, Enron and TREC07. In bigrams, the relationships between features are considered and a correlation matrix of tokens is developed; however, bigrams are created from the relationships of a token with only the neighbouring tokens as otherwise the correlation matrix would become too large causing high cost of processing and time. The Bayesian classifier was trained using 70% of the dataset by calculating the probability of bigrams as features. The parameters selected for evaluation of bigrams using each dataset are shown in Table 3.7.

Table 3.7: Parameter Sets used for Classification using Bigrams

Maximum Token Size	Spam cut off	Ham cut off
150	0.5	0.5
150	0.15	0.9
150	0.2	0.9
150	0.3	0.8
150	0.4	0.6
500	0.5	0.5
500	0.15	0.9
500	0.2	0.9
500	0.3	0.8
500	0.4	0.6
5000	0.5	0.5
5000	0.15	0.9
5000	0.2	0.9
5000	0.3	0.8
5000	0.4	0.6
10000	0.5	0.5
10000	0.15	0.9
10000	0.2	0.9
10000	0.3	0.8
10000	0.4	0.6
20000	0.5	0.5
20000	0.15	0.9

20000	0.2	0.9
20000	0.3	0.8
20000	0.4	0.6
25000	0.5	0.5
25000	0.15	0.9
25000	0.2	0.9
25000	0.3	0.8
25000	0.4	0.6

Results:

The classification results of the Bayesian classifier evaluated on 30% of the messages from each of the dataset are reported below:

For bigrams with 150 tokens, the classification results showed an FP rate <1% for three datasets-Ling-spam, Enron (except for 0.5 threshold) and TREC07 for all threshold values. For the PU1 dataset an FP rate between 1-4.3% resulted. The Grey rate was the lowest for the Enron dataset and range between 1.5-8% for the remaining three datasets as shown in Figure 3.14.

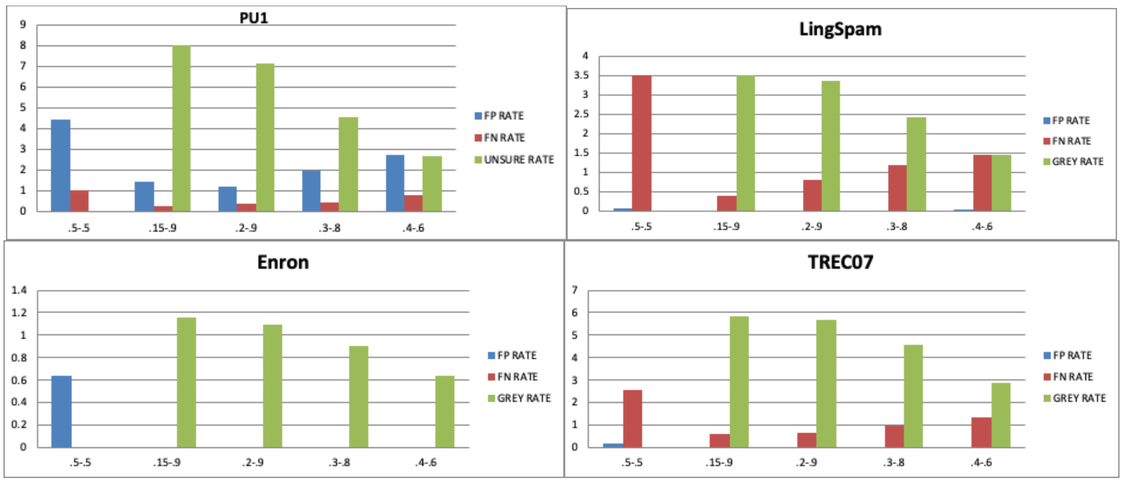


Figure 3.14: Test Results for all four datasets for Bigrams with token size of 150 for several threshold sets

For token size of 500 for bigrams (shown in Figure 3.15), the classifier performance was similar to the token size of 150 although the grey rates were lower for the Ling-spam and TREC07 datasets and the FP rates were lower for the PU1 and Enron datasets. Overall, having additional 250 bigram tokens did not change the performance.

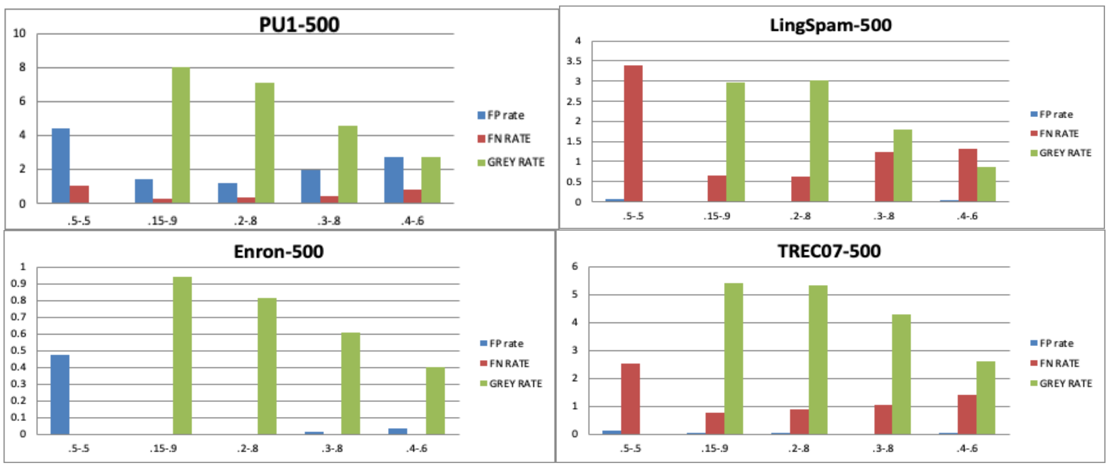


Figure 3.15: Test Results for all dataset for Bigrams with token size of 500 for several threshold sets

Selecting 5000 significant tokens for classification shows similar results for FP, FN and grey rates (Figure 3.16) for all four datasets as other token sizes.

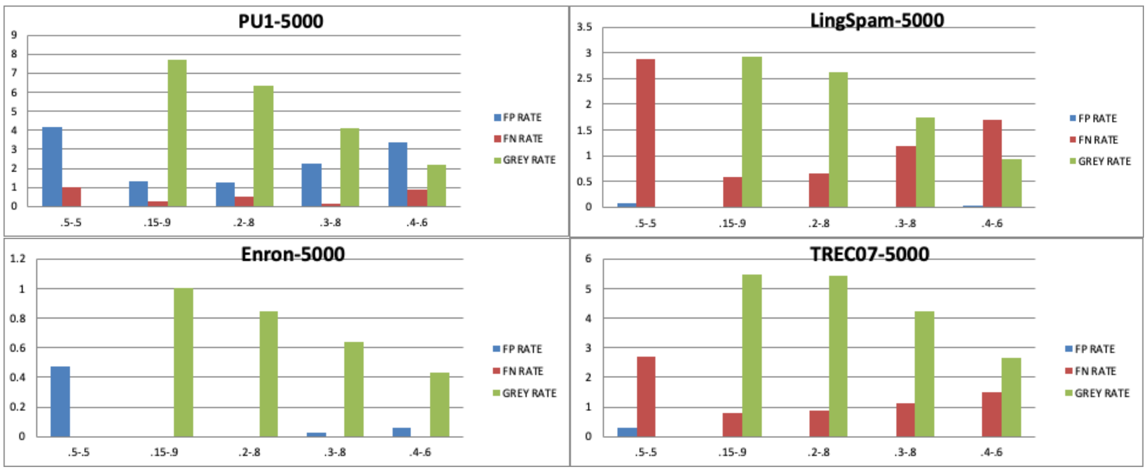


Figure 3.16: Test Results for all dataset for Bigrams with token size of 5000 for several threshold sets

It is, thus, noted that further increasing the token size does not make an impact on the outcome produced by the classifier as shown in Figures 3.17, 3.18 and 3.19, and the results are comparative and close.

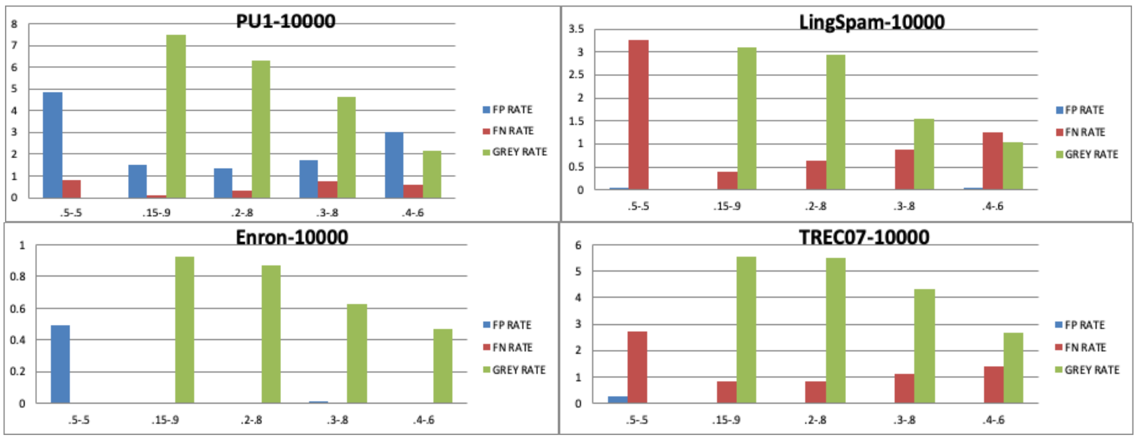


Figure 3.17: Test Results for all dataset for Bigrams with token size of 10000 for several threshold sets

With a token size of 10,000, FN and FP rates reduced slightly for far apart threshold values; however, they increased for 0.5-0.5 threshold values. The Grey rate remained comparable to smaller token sizes for all datasets.

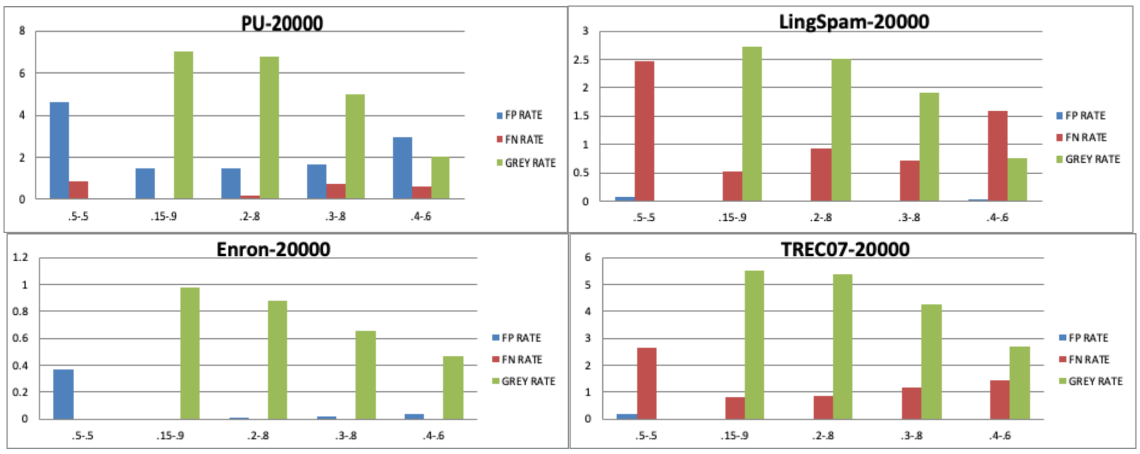


Figure 3.18: Test Results for all dataset for Bigrams with token size of 20000 for several threshold sets

By increasing the token size further to 20,000 as shown in Figure 3.18, FP and the Grey rate remained similar to lower token sizes used as features. The FN rate reduced slightly which means that spam detection efficiency increased.

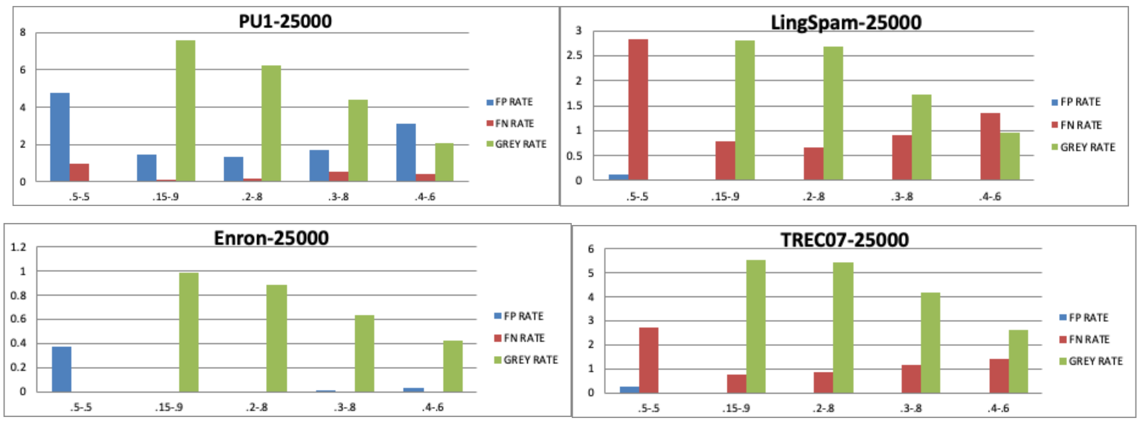


Figure 3.19: Test Results for all dataset for Bigrams with token size of 25000 for several threshold sets

For bigrams, the largest token size used in the experiment was 25000, with values of FP, FN rate and grey rates similar to 20000 bigram tokens used as features. Hence, this demonstrates that token size has no significant impact on classifier performance.

Table 3.8 show the comparative metrics for the performance of classifiers for all parameters. The FN, FP and grey rate are very similar for all token sizes. Increases in token sizes simultaneously increase the complexity of the correlation matrix; hence, if the classifier provides similar outcomes for smaller token sizes in terms of accuracy of spam or ham output class determination for smaller token size, then selecting a smaller token size for bigrams would be advisable.

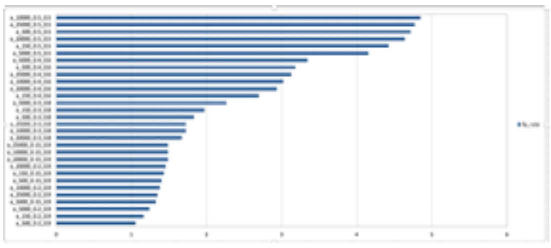
Table 3.8: Results for all Parameter Sets for all dataset for Bigrams

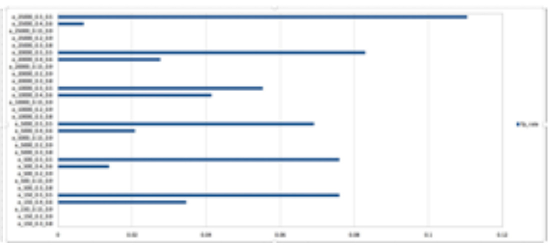
(on next page)

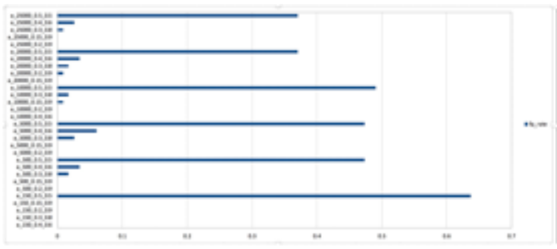
MAX	HAM	SPAM	FP RATE	FN RATE	GREY RATE	COST	FP RATE	FN RATE	GREY RATE	COST	FP RATE	FN RATE	GREY RATE	COST	FP RATE	FN RATE	GREY RATE	COST
TOKENS	CUT OFF	CUT OFF	PU1	PU1	PU1		LINGSPAM	LINGSPAM	LINGSPAM		ENRON	ENRON	ENRON		TREC07	TREC07	TREC07	
150	0.5	0.5	4.425916	1.005508	0	2.414059	0.075996	3.495211	0	0.73704	0.63683305	0	0	0.318417	0.135701	2.527517	0	0.573354
150	0.15	0.9	1.431279	0.243056	8.028938	3.172932	0	0.380987	3.515142	1.13074	0	0	1.158894	0.347668	0	0.588928	5.82161	1.864268
150	0.2	0.9	1.159408	0.346743	7.105764	2.780782	0	0.795977	3.347521	1.163452	0	0	1.094666	0.3284	0	0.641335	5.685381	1.833881
150	0.3	0.8	1.972553	0.416188	4.557861	2.436872	0	1.175287	2.419351	0.960863	0	0	0.906171	0.271851	0	0.961663	4.546829	1.556381
150	0.4	0.6	2.696609	0.797653	2.684308	2.313128	0.034578	1.453784	1.446352	0.741951	0	0	0.635297	0.190589	0	1.309491	2.856463	1.118837
			2.337153	0.56183	4.475374	2.623555	0.022115	1.460249	2.145673	0.946809	0.12736661	0	0.759006	0.291385	0.02714	1.205787	3.782056	1.389344
500	0.5	0.5	4.718407	1.03999	0	2.567202	0.076015	3.395354	0	0.717078	0.47332186	0	0	0.236661	0.109253	2.51374	0	0.557374
500	0.15	0.9	1.402365	0.173611	7.157493	2.883153	0	0.657567	2.961118	1.019849	0	0	0.941078	0.282323	0.044679	0.745832	5.410273	1.794588
500	0.2	0.9	1.052165	0.45091	6.598186	2.59572	0	0.623084	3.024297	1.031906	0	0	0.816811	0.245043	0.044705	0.879877	5.311599	1.791808
500	0.3	0.8	1.832757	0.485632	4.24543	2.287134	0	1.244971	1.803423	0.790021	0.0172117	0	0.610165	0.191655	0	1.062057	4.298144	1.501855
500	0.4	0.6	3.18018	0.555316	1.758501	2.228704	0.013812	1.316092	0.875756	0.532851	0.03442341	0	0.402122	0.137848	0.018657	1.414402	2.595782	1.070943
			2.437175	0.541092	3.951922	2.512382	0.017965	1.447414	1.732919	0.818341	0.10499139	0	0.554035	0.218706	0.043459	1.323181	3.52316	1.343314
5000	0.5	0.5	4.157525	1.040469	0	2.286856	0.06908	2.873324	0	0.609205	0.47332186	0	0	0.236661	0.288963	2.702055	0	0.684893
5000	0.15	0.9	1.322735	0.277059	7.692125	3.024417	0	0.58932	2.915972	0.992656	0	0	1.00391	0.301173	0	0.802971	5.469775	1.801527
5000	0.2	0.9	1.240487	0.520354	6.338964	2.626004	0	0.658285	2.616199	0.916517	0	0	0.844736	0.253421	0	0.892673	5.434281	1.808819
5000	0.3	0.8	2.265481	0.138889	4.097644	2.389812	0	1.175287	1.739673	0.756959	0.02581756	0	0.639486	0.204755	0	1.113718	4.222922	1.48962
5000	0.4	0.6	3.347302	0.90182	2.201032	2.514324	0.020737	1.694684	0.933367	0.629315	0.06024096	0	0.430047	0.159135	0	1.49332	2.638211	1.090127
			2.466706	0.575718	4.065953	2.568283	0.017963	1.39818	1.641042	0.78093	0.11187608	0	0.583636	0.231029	0.057793	1.400948	3.553038	1.374997
10000	0.5	0.5	4.847719	0.797414	0	2.583342	0.055315	3.251437	0	0.677945	0.49053356	0	0	0.245267	0.284714	2.746325	0	0.691622
10000	0.15	0.9	1.484749	0.104167	7.489854	3.010164	0	0.381226	3.088961	1.002933	0.00860585	0	0.922927	0.281181	0	0.831806	5.570283	1.837446
10000	0.2	0.9	1.374455	0.346743	6.321286	2.652962	0	0.623324	2.937957	1.006052	0	0	0.869869	0.260961	0	0.853445	5.516399	1.825609
10000	0.3	0.8	1.725375	0.762452	4.62602	2.402984	0	0.865182	1.538146	0.63448	0.0172117	0	0.628316	0.197101	0	1.104388	4.326348	1.518782
10000	0.4	0.6	3.020212	0.590038	2.184778	2.283547	0.041465	1.248324	1.048892	0.585065	0	0	0.471935	0.141581	0	1.423603	2.674614	1.087105
			2.490502	0.520163	4.124388	2.5866	0.019356	1.273898	1.722791	0.781295	0.10327022	0	0.578609	0.225218	0.056943	1.391913	3.617529	1.392113
20000	0.5	0.5	4.64286	0.832854	0	2.488001	0.082921	2.458094	0	0.533079	0.37005164	0	0	0.185026	0.190796	2.664007	0	0.6282
20000	0.15	0.9	1.483587	0.069444	7.038479	2.867226	0	0.520115	2.725639	0.921715	0	0	0.973192	0.291958	0	0.81336	5.51577	1.817403
20000	0.2	0.9	1.458597	0.207854	6.768146	2.801313	0	0.934626	2.517252	0.942101	0.00860585	0	0.875454	0.266939	0	0.86951	5.358677	1.781505
20000	0.3	0.8	1.673795	0.76341	5.006964	2.491669	0	0.727011	1.917761	0.720731	0.0172117	0	0.654845	0.205059	0	1.152701	4.271434	1.51197
20000	0.4	0.6	2.936356	0.623324	2.045004	2.206344	0.027643	1.58932	0.76033	0.559785	0.03442341	0	0.467746	0.157536	0	1.452657	2.702491	1.101279
			2.439039	0.499377	4.171719	2.570911	0.022113	1.245833	1.584196	0.735482	0.08605852	0	0.594247	0.221303	0.038159	1.390447	3.569674	1.368071
25000	0.5	0.5	4.774061	0.970785	0	2.581188	0.110574	2.835489	0	0.622385	0.37005164	0	0	0.185026	0.257384	2.693587	0	0.667409
25000	0.15	0.9	1.485624	0.138889	7.5902	3.04765	0	0.795259	2.805522	1.000708	0	0	0.988551	0.296565	0	0.778321	5.549466	1.820504
25000	0.2	0.9	1.349041	0.207854	6.232843	2.585944	0	0.657567	2.678305	0.935005	0	0	0.889416	0.266825	0	0.876668	5.429015	1.804038
25000	0.3	0.8	1.7277	0.519875	4.399115	2.28756	0	0.900144	1.716599	0.695008	0.00860585	0	0.629712	0.193217	0	1.137279	4.182345	1.482159
25000	0.4	0.6	3.128021	0.416667	2.077791	2.270681	0.006916	1.352251	0.9567	0.560918	0.02581756	0	0.417481	0.138153	0	1.423168	2.638438	1.076165
			2.492889	0.450814	4.05999	2.554605	0.023498	1.308142	1.631425	0.762805	0.08089501	0	0.585032	0.215957	0.051477	1.381805	3.559853	1.370055

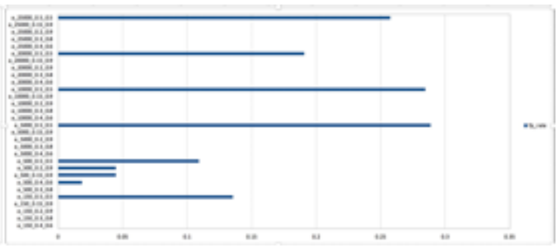
Figure 3.20 show the cost function and FP rate comparison for all datasets with all selected token sizes and threshold values for bigrams to identify the best performing combination of token size and threshold value.

FP Rate

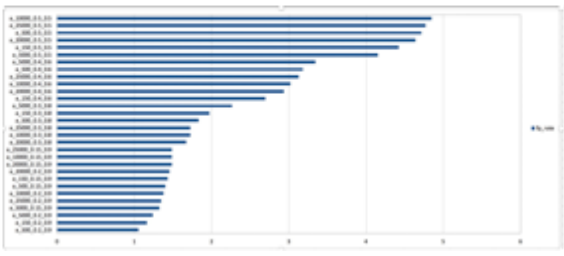

PU1

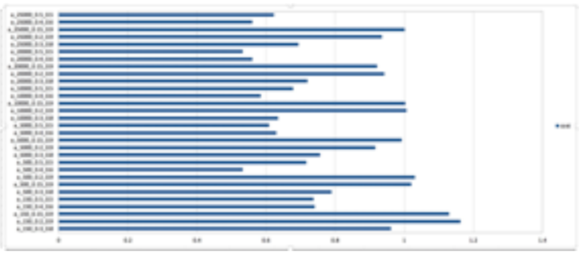

Ling-spam

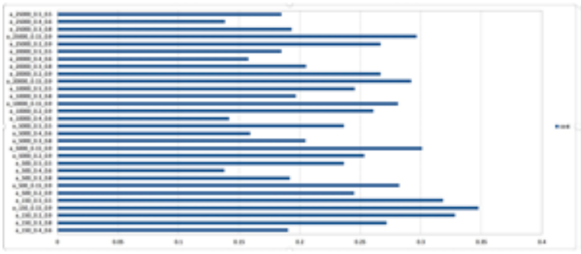

Enron


TREC07

Cost


PU1


Ling-spam


Enron

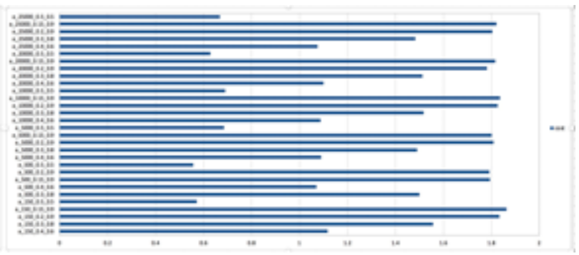

TREC07

Figure 3.20: FP Rate and Cost Function for All Four Datasets for Bigrams for All Parameter Values

The above comparison shows that increasing token sizes did not significantly contribute to the classifier performance for bigrams. Analysing the values and graphs from Table 3.8 and Figure 3.20, it was found that classifier performance is inferior for close threshold values although cost is lower since those threshold values lead to no or low greys whereas for far apart threshold values the classifier performs well with high cost due to high numbers of greys. Hence, for bigrams, far apart threshold values of 0.15 and 0.9 are chosen as best performing parameters for 150 token size.

Experiment 2.3: Trigrams with Varying Feature Set (Token) Size

To determine if correlation between three consecutive tokens impacts on the learning model and its capability to perform better in classification, trigrams were considered in this experiment, a correlation matrix was developed with the trigrams for each message, probability calculated and stored to be used for classification later.

Method: In this experiment, trigrams with varying significant token sizes of 150, 500, 5000, 10000, 20000, 25000 and 50000 for each of the threshold value sets given in table 3.1 was selected for each dataset. Only neighbouring tokens were considered to form the trigrams. For each trigram constructed from ham and spam datasets, the probability of the token was calculated and learned by the filter model. When a new message arrives, trigram features are extracted, probabilities calculated, and most the significant tokens are used to perform classification. The total number of parameters tested for trigrams for each dataset - PU1, Ling-spam, Enron and TREC07 is shown in Table 3.9.

Table 3.9: Parameter Sets for each dataset for Trigrams

Max Tokens	Spam cut off	Ham cut off
150	0.5	0.5
150	0.15	0.9
150	0.2	0.9
150	0.3	0.8
150	0.4	0.6
500	0.5	0.5
500	0.15	0.9
500	0.2	0.9
500	0.3	0.8
500	0.4	0.6
5000	0.5	0.5
5000	0.15	0.9
5000	0.2	0.9
5000	0.3	0.8
5000	0.4	0.6
10000	0.5	0.5
10000	0.15	0.9
10000	0.2	0.9
10000	0.3	0.8
10000	0.4	0.6

20000	0.5	0.5
20000	0.15	0.9
20000	0.2	0.9
20000	0.3	0.8
20000	0.4	0.6
25000	0.5	0.5
25000	0.15	0.9
25000	0.2	0.9
25000	0.3	0.8
25000	0.4	0.6
50000	0.5	0.5
50000	0.15	0.9
50000	0.2	0.9
50000	0.3	0.8
50000	0.4	0.6

Results:

Figures 3.21 to 3.27 show that the FP rate was not impacted by increasing the token size. The FN rate increased slightly by increasing the token size across all datasets.

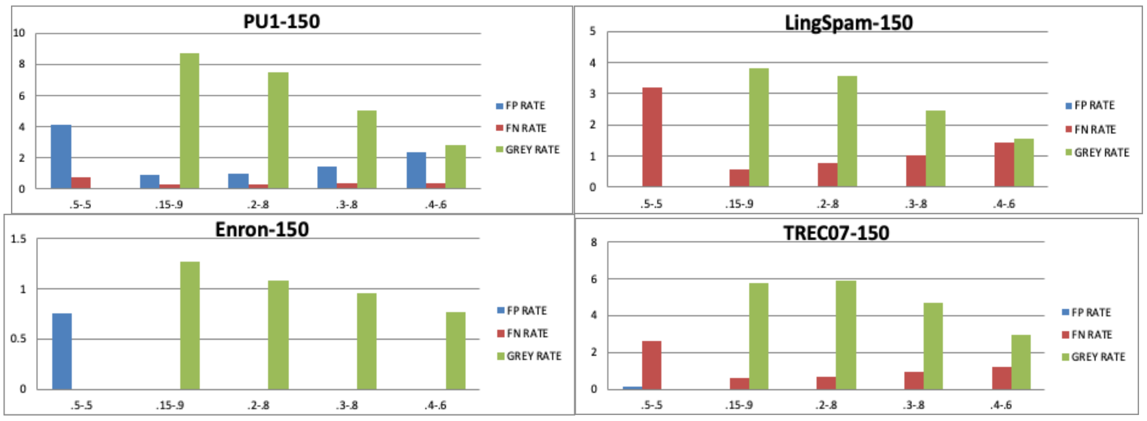


Figure 3.21: Test Results for all dataset for Trigrams with token size of 150 for several threshold sets

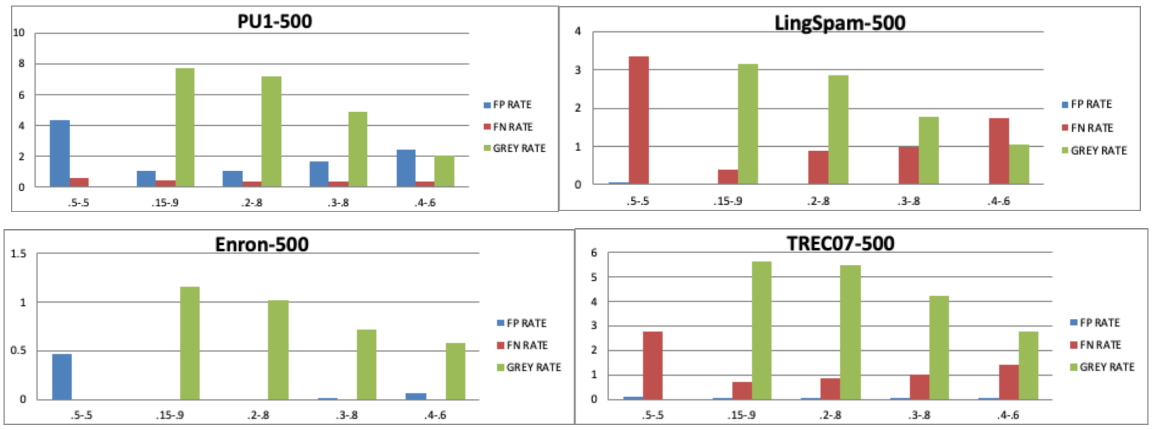


Figure 3.22: Test Results for all dataset for Trigrams with token size of 500 for several threshold sets

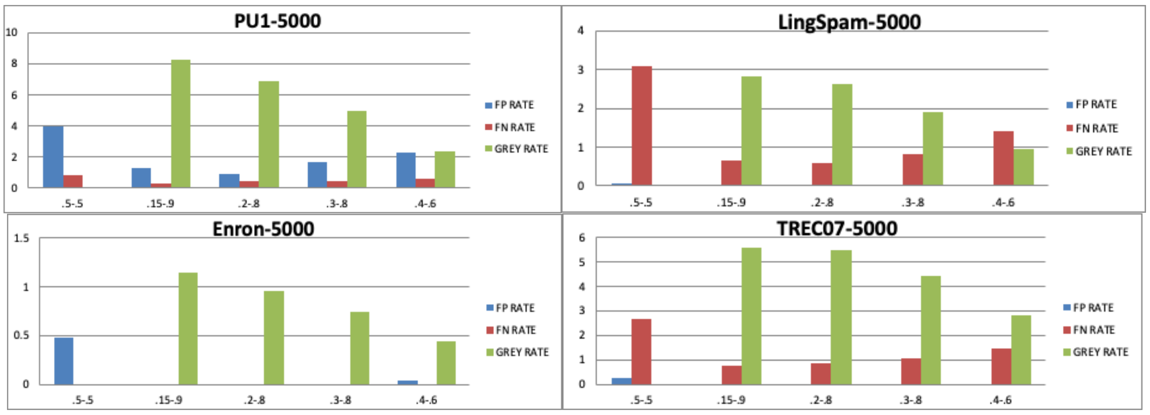


Figure 3.23: Test Results for all dataset for Trigrams with token size of 5000 for several threshold sets

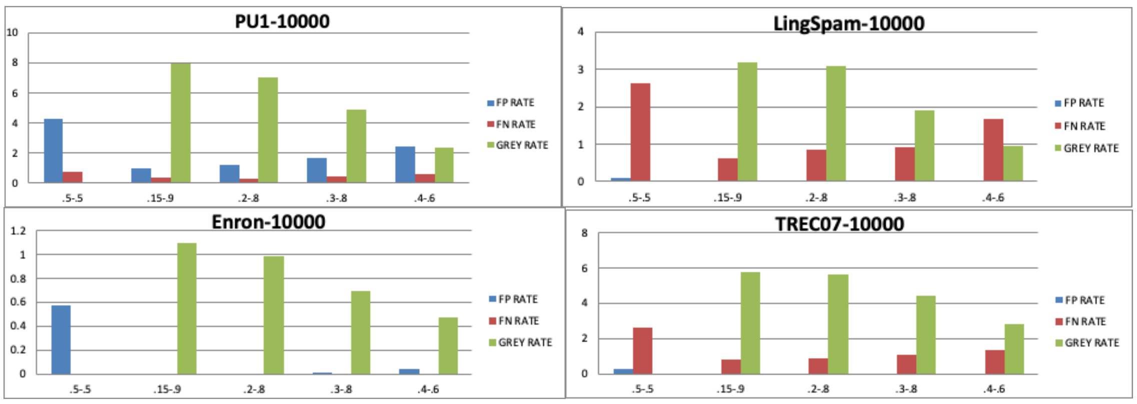


Figure 3.24: Test Results for all dataset for Trigrams with token size of 10000 for several threshold sets

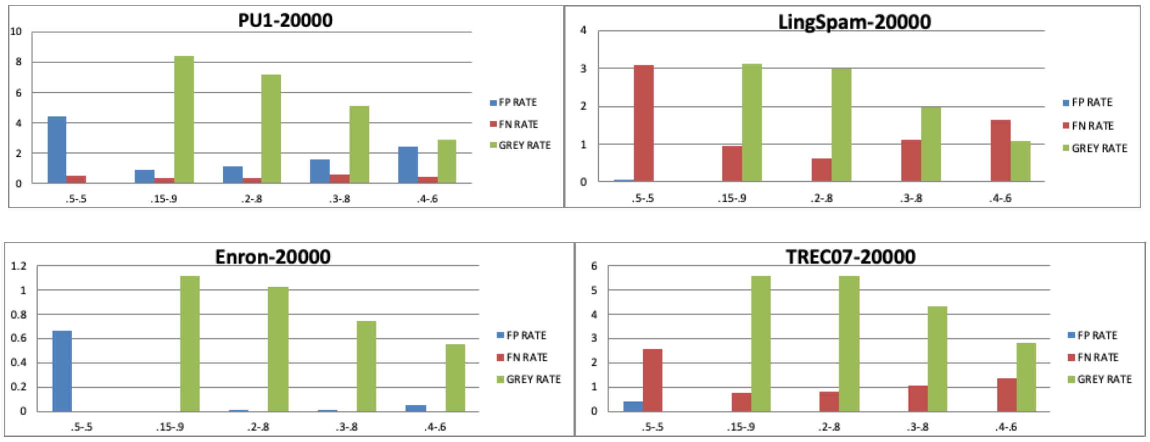


Figure 3.25: Test Results for all dataset for Trigrams with token size of 20000 for several threshold sets

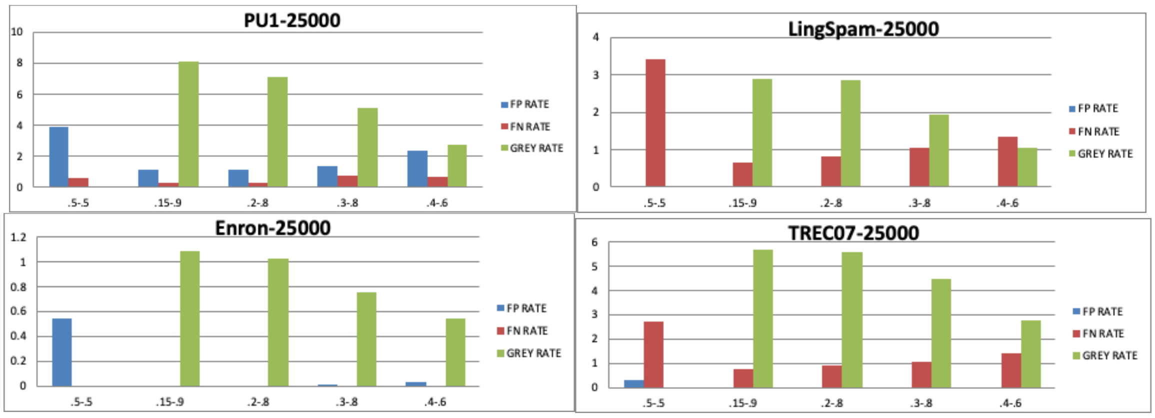


Figure 3.26: Test Results for all dataset for Trigrams with token size of 25000 for several threshold sets

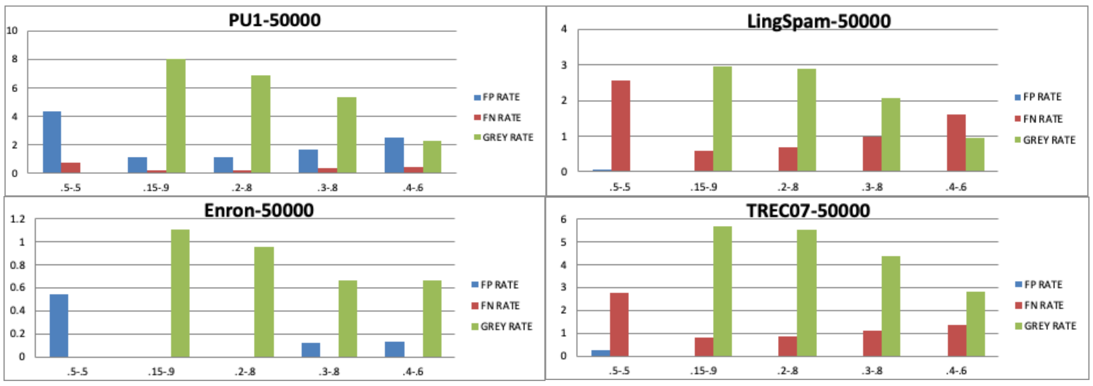


Figure 3.27: Test Results for all dataset for Trigrams with token size of 50000 for several threshold sets

From the results compared in Table 3.10, it was observed that the FP rate increased slightly for all datasets as the token size increased. For the FN rate no pattern was detected though the performance of the classifier produced comparative results for all parameters. The Grey rate skewed towards higher values for far apart threshold values although the FP rate showed good performance for those thresholds. The cost function confirms comparative performance of the classifier as the difference in value is negligible.

Table 3.10: Results for all Parameter Sets for all dataset for Trigrams

MAX TOKENS	HAM CUTOFF	SPAM CUTOFF	FP RATE PU1	FN RATE PU1	GREY RATE PU1	COST	FP RATE LINGSPAM	FN RATE LINGSPAM	GREY RATE LINGSPAM	COST	FP RATE ENRON	FN RATE ENRON	GREY RATE ENRON	COST	FP RATE TREC07	FN RATE TREC07	GREY RATE TREC07	COST
150	0.5	0.5	4.148358	0.728448	0	2.219869	0.034559043	3.188457854	0	0.654971	0.757314974	0	0	0.378657	0.156297559	2.643325	0	0.606814
150	0.15	0.9	0.919224	0.312021	8.682133	3.126656	0	0.554118774	3.831381213	1.260238	0	0	1.263614	0.379084	0	0.572218	5.798838	1.854095
150	0.2	0.9	0.998405	0.277299	7.493861	2.80282	0	0.796695402	3.56707958	1.229463	0	0	1.075119	0.322536	0	0.686223	5.877857	1.900602
150	0.3	0.8	1.483737	0.416188	5.052018	2.340711	0	1.038314176	2.46597759	0.947456	0	0	0.950852	0.285256	0	0.933398	4.664807	1.586121
150	0.4	0.6	2.39989	0.381466	2.821509	2.122691	0.027662517	1.454262452	1.549674755	0.769586	0	0	0.770734	0.23122	0	1.233538	2.967943	1.137091
			1.989923	0.423084	4.809904	2.522549	0.012444312	1.406369732	2.282822628	0.972343	0.151462995	0	0.812064	0.319351	0.031259512	1.21374	3.861889	1.416944
500	0.5	0.5	4.36924	0.589799	0	2.30258	0.055277301	3.35967433	0	0.699574	0.464716007	0	0	0.232358	0.119756941	2.742612	0	0.608401
500	0.15	0.9	1.079047	0.416667	7.735621	2.943543	0	0.380507663	3.139934554	1.018082	0	0	1.161687	0.348506	0.009578544	0.726515	5.641066	1.842412
500	0.2	0.9	1.078909	0.381944	7.143077	2.758767	0	0.899185824	2.856205443	1.036699	0	0	1.012287	0.303686	0.009009009	0.868898	5.475826	1.821032
500	0.3	0.8	1.6475	0.346743	4.874899	2.355568	0	0.968869732	1.76307289	0.722696	0.008605852	0	0.713488	0.218349	0.045106393	1.011495	4.240571	1.497024
500	0.4	0.6	2.451613	0.416188	2.076459	1.931982	0.013831259	1.727729885	1.048269338	0.666942	0.060240964	0	0.576655	0.203117	0.009191176	1.383174	2.748396	1.105749
			2.125262	0.430268	4.366011	2.458488	0.013821712	1.467193487	1.761496445	0.828798	0.106712565	0	0.692823	0.261203	0.038528413	1.346539	3.621172	1.374923
5000	0.5	0.5	3.964109	0.865661	0	2.155187	0.069099008	3.08045977	0	0.650641	0.473321859	0	0	0.236661	0.266663288	2.679514	0	0.669235
5000	0.15	0.9	1.269125	0.3125	8.285976	3.182855	0	0.657806513	2.8295319	0.980421	0	0	1.146328	0.343898	0	0.764832	5.564806	1.822408
5000	0.2	0.9	0.889719	0.449952	6.89606	2.603668	0	0.58908046	2.639213944	0.90958	0	0	0.95504	0.286512	0	0.863716	5.504816	1.824188
5000	0.3	0.8	1.672926	0.450431	4.928781	2.405183	0	0.831417625	1.901150657	0.736629	0	0	0.746998	0.224099	0	1.076187	4.434441	1.54557
5000	0.4	0.6	2.29251	0.588841	2.365816	1.973768	0.013831259	1.419300766	0.944987025	0.574272	0.034423408	0	0.435633	0.147901	0	1.452391	2.806895	1.132547
			2.017678	0.533477	4.495327	2.464132	0.016586053	1.315613027	1.662976705	0.770309	0.101549053	0	0.6568	0.247814	0.053332658	1.367328	3.662192	1.398789
10000	0.5	0.5	4.291657	0.797414	0	2.305311	0.089855421	2.628591954	0	0.570646	0.576592083	0	0	0.288296	0.23613678	2.602459	0	0.63856
10000	0.15	0.9	1.024269	0.414751	7.937427	2.976313	0	0.622605364	3.184909113	1.079994	0	0	1.098855	0.329657	0	0.79408	5.732188	1.878472
10000	0.2	0.9	1.240192	0.277778	7.052375	2.791364	0	0.864463602	3.081821386	1.097439	0	0	0.985758	0.295727	0	0.882272	5.61527	1.861035
10000	0.3	0.8	1.643999	0.449713	4.877841	2.375294	0	0.901340996	1.895846496	0.749022	0.008605852	0	0.691148	0.211647	0	1.043837	4.441966	1.541357
10000	0.4	0.6	2.42692	0.588602	2.395145	2.049724	0.013831259	1.662356322	0.950733785	0.624607	0.04302926	0	0.469143	0.162257	0	1.374369	2.833804	1.125015
			2.125407	0.505651	4.452558	2.499601	0.020737336	1.335871648	1.822662156	0.824342	0.125645439	0	0.648981	0.257517	0.047227356	1.339403	3.724645	1.408888
20000	0.5	0.5	4.422999	0.520115	0	2.315522	0.048342568	3.081657088	0	0.640503	0.662650602	0	0	0.331325	0.396746372	2.579626	0	0.714298
20000	0.15	0.9	0.890733	0.345785	8.435692	3.045231	0	0.9348659	3.110677373	1.120176	0	0	1.118403	0.335521	0	0.745901	5.5655	1.81883
20000	0.2	0.9	1.107243	0.416188	7.208642	2.799451	0	0.623563218	2.990842246	1.021965	0.008605852	0	1.023457	0.31134	0	0.810814	5.582458	1.8369
20000	0.3	0.8	1.564962	0.58932	5.12911	2.439078	0	1.107279693	1.970652737	0.812652	0.008605852	0	0.745602	0.227983	0	1.03442	4.345699	1.510594
20000	0.4	0.6	2.4551	0.485872	2.866496	2.184673	0.006906077	1.626676245	1.077315987	0.651983	0.051635112	0	0.554314	0.192112	0	1.350695	2.822901	1.117009
			2.088207	0.471456	4.727988	2.556791	0.011049729	1.474808429	1.829897669	0.849456	0.146299484	0	0.688355	0.279656	0.079349274	1.304291	3.663311	1.399526
25000	0.5	0.5	3.881731	0.588362	0	2.058538	0.034539939	3.425766284	0	0.702423	0.542168675	0	0	0.271084	0.298801278	2.703939	0	0.690188
25000	0.15	0.9	1.105644	0.312021	8.128798	3.053866	0	0.658524904	2.898796825	1.001344	0	0	1.090478	0.327143	0	0.765707	5.664889	1.852608
25000	0.2	0.9	1.133246	0.311542	7.125282	2.766516	0	0.829741379	2.851100586	1.021278	0	0	1.022061	0.306618	0	0.913364	5.596592	1.86165
25000	0.3	0.8	1.403818	0.72773	5.115292	2.382043	0	1.038793103	1.925000565	0.785259	0.008605852	0	0.751187	0.229659	0	1.071883	4.479275	1.558159
25000	0.4	0.6	2.377803	0.693966	2.77881	2.161338	0	1.349856322	1.042893223	0.582839	0.034423408	0	0.538956	0.178898	0	1.409599	2.774618	1.114305
			1.980449	0.526724	4.629636	2.48446	0.006907988	1.460536398	1.74355824	0.818629	0.117039587	0	0.680536	0.262681	0.059760256	1.372898	3.703075	1.415382
50000	0.5	0.5	4.366173	0.797414	0	2.342569	0.055267749	2.559626437	0	0.539559	0.542168675	0	0	0.271084	0.249256927	2.772192	0	0.679067
50000	0.15	0.9	1.159406	0.207615	8.008364	3.023735	0	0.588362069	2.938640472	0.999265	0	0	1.10444	0.331332	0	0.787601	5.666799	1.85756
50000	0.2	0.9	1.158384	0.207615	6.865515	2.680369	0	0.691810345	2.885564024	1.004031	0	0	0.959229	0.287769	0	0.856068	5.514677	1.825617
50000	0.3	0.8	1.699365	0.347222	5.309268	2.511907	0	0.967193487	2.079237226	0.81721	0.125502008	0	0.667097	0.26288	0	1.106637	4.361877	1.52989
50000	0.4	0.6	2.534299	0.45067	2.304012	2.048487	0	1.590996169	0.961558641	0.606667	0.130731258	0	0.663768	0.264496	0	1.348141	2.831114	1.118963
			2.183525	0.402107	4.497432	2.521414	0.01105355	1.279597701	1.773000073	0.793346	0.159680388	0	0.678907	0.283512	0.049851385	1.374128	3.674893	1.402219

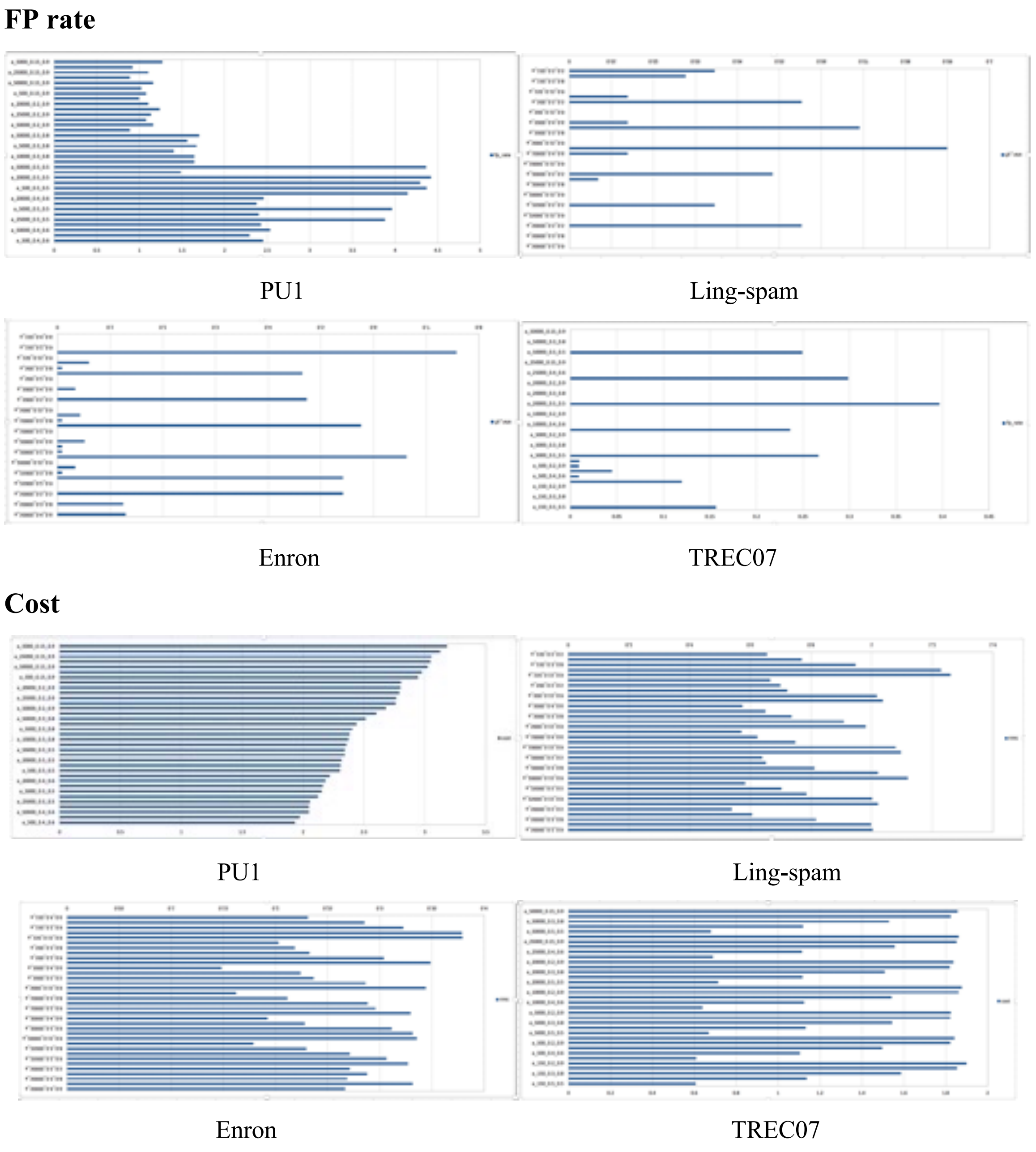


Figure 3.28: FP rate and cost function for all four datasets for Trigrams for all parameter values

3.9 Analysis of Results and Discussion – Modified Bayesian Classifier

Initial results indicated inconsistent behaviour among the datasets, especially for the PU1. The value of FP, FN and grey varied slightly for all 5 datasets. Different data sets showed different optimum values for cut-offs and token sizes. This may mean that each data set differs since data sets belong to different times, probably have different styles of both spam and ham emails belonging to different authors and spam designs. From the results for unigrams, bigrams and trigrams it is noted that FP and FN rates consistently increased consistent with the widening of the gap between spam and ham cut offs. For median values of the threshold, it is observed that the classifier performed with high accuracy for low FP rates.

In general, it is noticed that FN and grey rates skew towards higher values which means the accuracy of the classifier is low. From the overall results, it is noticed that the following models had the best scores amongst all of parameters

For the unigram model 50, 0.4, 0.6, (and (50, 0.5, 0.5))

For the bigrams model 20000, 0.4, 0.6, (and (20000, 0.5, 0.5))

For the trigrams model 5000, 0.4, 0.6 (and (5000, 0.5, 0.5))

Since the FP rate for the best performing models for bigrams and trigrams are comparable, it is important to consider the execution cost. It transpired that the model with 0.4 and 0.6 cut-offs and the same model with 0.5 and 0.5 cut-offs had similar execution times. For example, the times for 20 iterations of 3393 messages are:

unigram: 302 seconds

bigrams: 391 seconds

trigrams: 476 seconds

A general increase in time of 29% for bigrams and 57% for trigrams was observed compared to unigram models.

When computing the average time for one email (divide seconds with 20*3393), the same percentage increase was observed. However, considering that one email is scored as far below 1 second, it could be argued that this performance penalty is acceptable if emails are "coming slowly" to the inbox and processing requires 60% more time (although that is still much less than a second).

In this light, it can be concluded that training data sets have a significant impact on choice of parameters for classification (K. Bajaj & Pieprzyk, 2014). Further, it is deduced that increasing the token size for bigrams does not increase the performance of the classifier. For token sizes of 150-200 for unigrams and bigrams, there is similar performance. If the FP rate is low, that means high accuracy of the model; however, the values of the FN and the grey rates were as high as 6% and 9% respectively. Hence, it is concluded that further research with unigrams and bigrams of token size 150 to improve the FN rate and eliminate greys would certainly contribute to improving the performance of the Bayesian classifier.

The next chapter develops a framework for the inclusion of semantic and syntactic feature sets which can be fed into the classification filter which adds another layer of classification. The results of this addition upon classification are then tested across a series of experiments to determine positive and negative effects followed by a discussion.

Chapter 4

DMLEM-Dynamic Multi-Layer Ensemble model for high precision classification of Greys

Security in IT is like locking your house or car – it doesn't stop the bad guys, but if it's good enough they may move on to an easier target. — Paul Herbka

4.1 Overview

In the previous chapter, it was shown that Bayesian Classifier provides an additional level of email spam content filtering at the client side to detect the email spam that escaped the mail server. This classifier was designed using a Naïve Bayes classification algorithm where the default value of tokens is unigrams. It has been shown that email spam detection rates can be improved by using bigrams with little cost in terms of time, efficiency and computation. It works with numerous threshold values. Several experiments conducted on variety of datasets have demonstrated that this Bayesian classifier provides good results for obtaining a reasonable trade-off between false positives and false negatives, especially the false positive rate.

However, although the classifier achieves the best performance for low false positives and false negatives with far apart threshold values, it leads to higher numbers of unclassified emails, referred to as 'greys'. This requires substantial time commitment from the user to filter these greys and classify them manually to obtain high performance from the filter. Client-side email filtering is designed to be efficient and independent, with the purpose of reducing the impact of email spam such as loss of user productivity, time and cost.

Hence, in this chapter a multi-layer model is proposed that suggests an enhancement to the Bayesian Classifier - an additional layer of filtering for use as client-side filter. It is proposed to use techniques for, first, feature selection and then, classification of unclassified greys in the Bayesian Classifier into ham and spam adding a further supervised machine learning algorithm. The goal of **supervised learning** (predictive approach) is to learn from examples. This data is called 'training set', based on which an algorithm generalises patterns that correctly respond to all possible inputs. Suppose there is a labelled set of input-output pairs in a training set T with a given set of X training examples of input a which forms a vector of dimension T and an output b. Given the labelled training set T = $\{(a_i, b_i)\}_{i=1}^{X}$ that contains mapping between a and b, the aim is to identify a pattern. Here, each training input a_i may be, for example, the length and weight of an object, known as features or attributes. The training input could also represent an image, a sentence or a document such as an email message, etc. The output b_i may be anything in principle, but the common assumption is that it is a finite set of outputs such as b_i belongs to {s, t,….z} such as day or night known as categorical or even real value scalars, known as regression. The supervised algorithms used in the proposed model are correlation and regression technique, logistic regression, k-nearest neighbour and support vector machine.

Furthermore, this Bayesian Classifier only relies on text features and assumes that all the unigram features are independent and uncorrelated. During the feature selection process, it performs feature reduction by selecting 150 significant unigram tokens. Hence, selecting appropriate features with correct representation is critical to this classifier's performance.

Feature selection is essential to creating an efficient content-based filtering system taking into account the computation complexity and the classifier performance. One of the major challenges of content-based filtering is the relevance of features to email spam filtering and the redundancy between features. Selection of appropriate techniques and methods for feature selection can precisely determine the appropriate features and address these challenges. There are several methods used by researchers for selection of appropriate features from header and/or body of emails for content-based email spam filtering. However, there is limited evidence in research for feature selection with the use of syntactic features. Furthermore, to the best of our knowledge, there are no research papers using syntactic features for filtering email spam at the client side as additional layer of spam detection. J. Kim et al. (2007) used a Naive Bayes model to focus on URLs in the email messages instead of considering words in the filter they developed. As part of the learning process, the filter was updated with the messages that were classified. The filter was fed back only periodically for correctly classified messages but more regularly for incorrectly classified ones. A. K. Sharma & Yadav (2015) suggested a data mining technique for client-based filtering; however, no implementation for clients has been presented, proposed or mentioned as future work. This thesis presents a model that fills this gap in research.

In this chapter, for feature selection and classification of ham and spam emails, a BoW approach with term frequency using sparse features is proposed. The Term Frequency method provides tokenisation of a collection of text documents and builds a vocabulary of words converting the raw collection of text to a numerical vector representation of terms and n-grams which makes this easy to use this representation as features for classification in machine learning. BoW and the Term Frequency technique have been used for feature selection in email spam detection and achieved good results. However, they have not been used together and for syntactic feature selection. Here, these two techniques are proposed to be used for semantic and syntactic feature selection and email classification into two output classes as ham and spam. In further sections of the chapter, the implementation of these two techniques for selection of semantic and syntactic features and then discrimination of emails into legitimate ham and malicious spam is described. Here, the solution for feature relevance and reduction through the selection of structural features as syntactic features are explained as well as identification of groups of features which enable an efficient use of the Bayesian Classifier for client-side email spam filtering.

The selected features are evaluated on two experiments using an individual supervised machine learning algorithm, correlation and regression technique as part of proposed multilayer dynamic model (DMLM) as presented in Section 4.2 and results are presented in Section 4.6. There is evidence that suggests that ensembles tend to perform better than individual models. Therefore, the subsequent experiment develops a bagging ensemble of four supervised machine learning algorithms as stated earlier to perform evaluation in the additional layer of the proposed model. With this ensemble, the model becomes a dynamic multi-layer ensemble model (DMLEM) as presented in Section 4.7, the model is termed "Dynamic" since the model is updated in real time as soon as classification of the email message is complete. The outcome of the classification feeds back into the model generation process to update the learning dynamically as shown in figure 4.4. The results of experiments conducted in section 4.7 are presented in Section 4.8 along with a comparison with other ensemble models from current research although these latter models are designed for spam detection at the server side and are not multi-layer. Furthermore, the classification is performed with thresholds such that there is only one outcome of spam or ham. Hence, the proposed model is a novel technique for email spam detection. Section 4.9 contains a detailed analysis of results and discussion followed by a conclusion to the section.

4.2 Experiment 3: Dynamic Multi-layer email spam detection Model (DMLM) for Classification of Greys

A solution to spam emails remains elusive despite more than a decade of research efforts on spam filtering. As stated earlier, Naïve Bayesian algorithm-based content filtering has achieved a reasonable level of success among the spam detection mechanisms proposed in the past. Chapter

3 introduced the Bayesian classifier, a content filtering spam detection tool based on Naïve Bayesian classification using textual features.

This Bayesian classifier uses semantic text features to classify emails into three classes: spam, ham and grey. Grey is an email that lies between the two threshold values called the grey area. It is not classified as spam or ham as it contains features that belong to both. It was observed in Chapter 3 that the performance of the Bayesian classifier meets high expectations except for this third classification output class ‘grey. Grey emails lead the user to spend time in manually classifying those emails to determine if any important emails were incorrectly categorised as greys, adding to false positives. This manual classification is inefficient and causes loss to user in terms of time and money.

Hence, in this section of the chapter a dynamic multi-layer model (DMLM) is proposed that is superimposes onto the client-based Bayesian classifier - a second layer of filtering containing text (semantic) and non-textual (syntactic) features. This additional layer, shown as layer 2 in Figure 4.1, has three main objectives of eliminating the greys, to detect false positives that may have slipped through the Bayesian classifier, and to increase the accuracy of email spam detection.

This layer 2 is used to detect (1) email spam that was misclassified by the mail server as legitimate email message to arrive in the user’s email client and then also classified by the Bayesian classifier filter as ‘grey’ and (2) ham that is incorrectly classified by the Bayesian classifier as ‘greys’ or spam. This 2-layer filtering at client side is shown in Figure 4.1 where layer 1 is the Bayesian classifier and layer 2 is the additional layer that forms the DMLM.

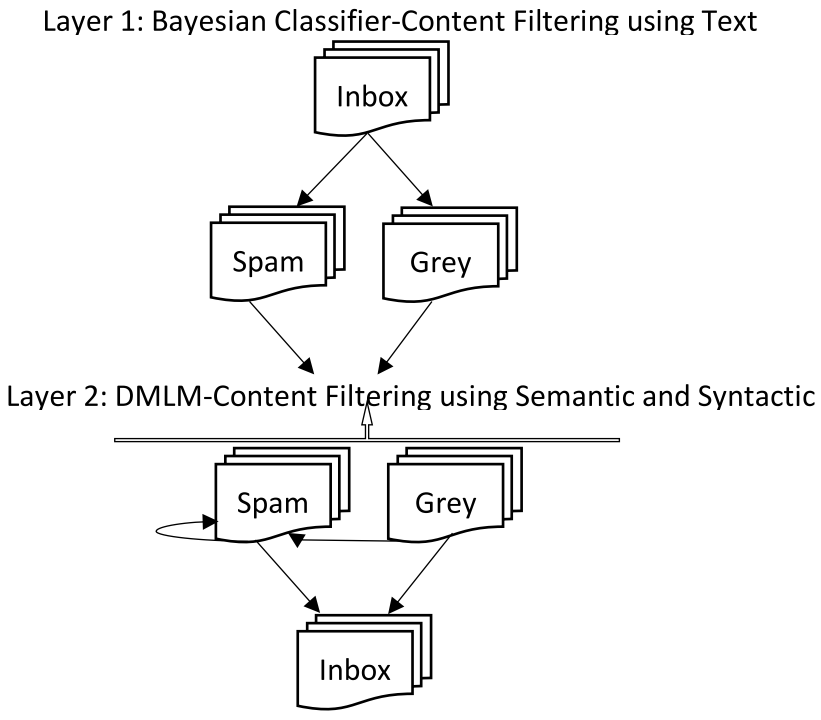

Layer 1: Bayesian Classifier-Content Filtering using Text
Inbox
Spam
Grey
Layer 2: DMLM-Content Filtering using Semantic and Syntactic
Spam
Grey
Inbox

Figure 4.1: 2-stage filtering at client side

This proposed model eliminates greys by re-classifying the grey messages. The second layer in the model performs the following classification steps (1) categorizes grey emails into clear spam and ham, where spam is moved to a spam folder and ham to the Inbox and (2) reclassifies spam messages into ham and spam, where ham in forwarded to the Inbox and spam stays in the spam folder. The objective of this model is two-fold, first to recover the loss of important emails as false positives in folders for greys and spam created by Bayesian classifier and second to improve the learning for enhanced classification performance accuracy. This multi-layer model improves the accuracy of classification. The experimental results of this model are encouraging. The framework for multi-layer email spam detection model is shown in Figure 4.2.

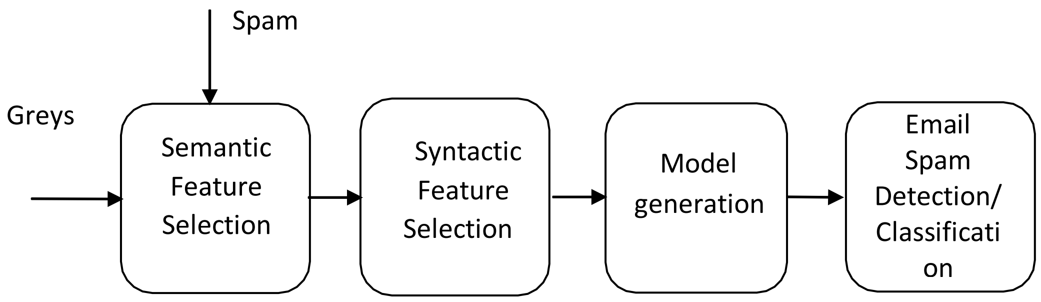

Spam
Greys
Semantic Feature Selection
Syntactic Feature Selection
Model generation
Email Spam Detection/ Classificati on

Figure 4.2: Framework for Multi-Layer Email Spam Detection Model

It has four stages - semantic feature selection, syntactic feature selection, model generation and email spam detection and classification. In content filtering, the raw feature set is composed of large numbers of features which lead to high computation cost in terms of power consumption and time for the classification process. The Bayesian classifier handles this by considering only 150 significant tokens. However, it has a limitation in that it does not categorise the emails into two the classes of ham and spam. The multi-layer spam detection model visualises the classification problem as two output class categorisations (ham and spam). Here, a count vectorizer method is used to extract features and then selected feature set is used to build a spare matrix. In the first stage, it builds an initial attribute set from semantic text features, then considers and selects syntactic non-text features. Several syntactic features such as numbers of link symbols, numbers of mis-spelt words and overuse of numeric characters provide significant information about emails. Finally, all the selected features are integrated into a new significant feature set as a frequency matrix. This is used for model generation and email spam detection and classification using the machine learning process. It then sends the feedback to the model for the learning process.

The first step is to collect and understand the data and its structure before preparing it for the algorithms to learn from. Before data can be used for learning, it must have certain conditions related to content, amount, format and presentation. Classifiers and algorithms in machine learning do not perform well if the data is of poor quality in terms of accuracy and sufficiency in terms of size. Hence data preparation involves gathering, pre-processing and representing the data in the right format. The format may take the form of attribute value vectors where attribute is the characteristic known as token and the value is the output also known as class. The next step consists of identification of features that represent this data and that will point to a particular type of output. Thereafter, the data is fed into statistical distribution via a model suited to text and numeric or image data. Machine learning algorithms are fed with the prepared data to build a model to be used for classifying novel inputs if these inputs follow the representation that is suitable. The model is then evaluated using the test harness. Data representation, feature selection and model training are the key processes.

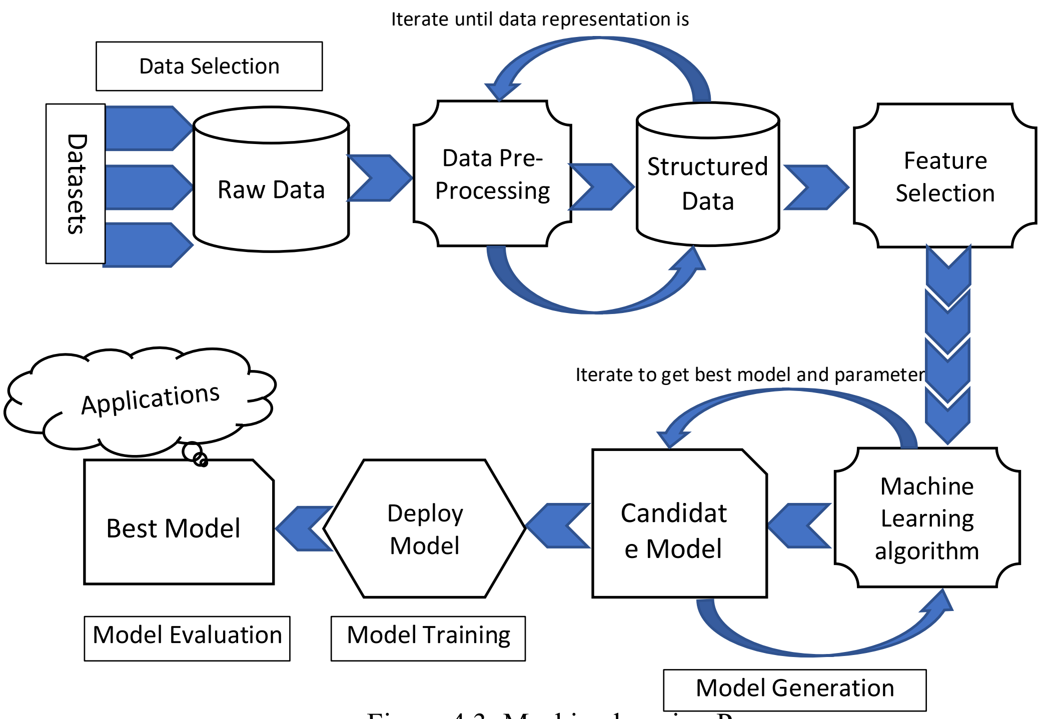


Figure 4.3: Machine learning Process

Data Selection: This step is important as the type and size of data chosen will affect the entire learning process of the model. The data should be representative of the domain for which the model is being developed. For supervised learning, the data contains input as well as output values; input values are independent, while outputs are dependent on input attributes.

Data Pre-processing: The data in its raw form is sometimes flawed and may not be suitable. It is important to ensure that errors are identified, cleaned, secured and governed. Errors may be empty spaces or even columns. The pre-processing of data involves addressing errors, as well as missing values and outliers.

Data Representation: refers to the form in which data is stored and processed for learning. Data must be prepared in the format that correctly represents populations or domains. Data may be skewed and relationships which represent imbalances may lead to bias. This stage also involves adjustment and manipulation such as normalisation, augmentation, de-duping of data so that final prepared data contain attributes to shape the dataset for the purpose. The focus is to analyse the input data and then to develop a suitable representation to learn from.

Feature Selection: During this process, important features are selected that assist in choosing appropriate data relevant to the problem domain to feed into the model. Also called attribute selection or variable selection, it is an automatic selection of characteristics that are most relevant to the prediction problem in question. It acts as a filter, aims to reduce the number of attributes by muting attributes that are not useful as some machine learning algorithms do not perform well in high dimensional spaces. This creates a simpler model that is easy to understand, is fast and has improved performance. The aim of feature selection is to aid in developing an accurate classification machine learning model for prediction. For the text categorisation problem, the most common features are characters, words and strings that convey a meaningful message.

Model Generation and Parameter Selection: This step involves the selection of a suitable model and associated parameters. There is no guidance for the choice of machine learning algorithm to build the model and so far, there is no machine learning algorithm that has been identified as superior. Some algorithms are more powerful than others, non-parametric, flexible as well as self-tuning but may be difficult to implement especially for large datasets where simpler models scale and perform better. In practice, that means that many algorithms need to be tested to select those that work for the problem in question with acceptable speed, performance, reliability and accuracy (Brownie, 2018).

AS for machine learning algorithms, there are no best machine learning algorithm parameters either. This means, the same approach needs to be applied to grid search and select the parameters that provide acceptable output for performance, reliability and accuracy.

Model Deployment and Training: Any algorithm needs to be trained to create the model. In this step, the model is initialised with features and corresponding labels, where the model identifies the patters in the example data that map these features onto the label. During this stage, the user may define the control parameters which can be adjusted by optimising performance using cross validation.

Model Evaluation: The model is evaluated to find the best performing algorithm and parameters. Finally, the accuracy of the learnt function is evaluated using the test dataset which is distinctively different from the dataset used for training this model.

The architecture for the Dynamic Multi-Layer Model (DMLM) is shown in Figure 4.4

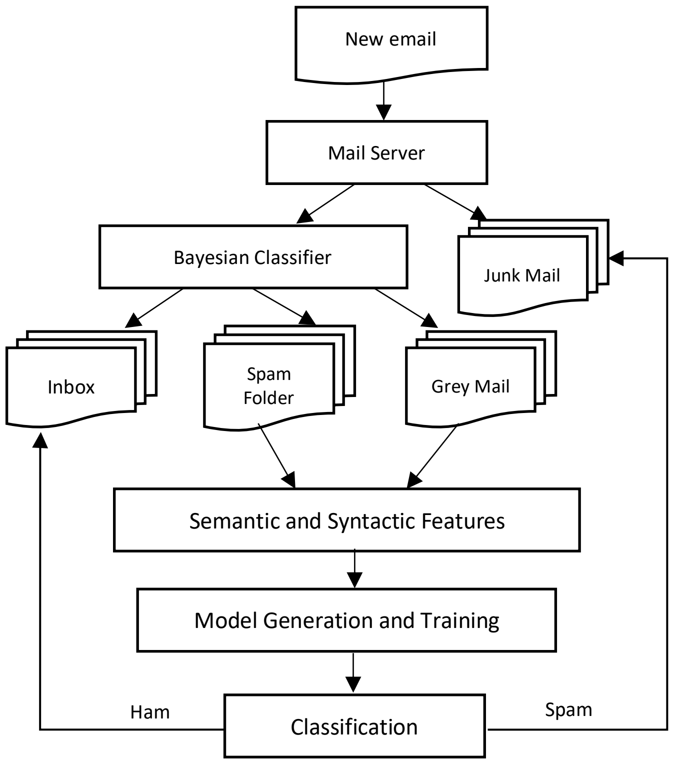


Figure 4.4: Architecture of DMLM

4.3 Semantic Feature Selection

Text Features help the user make sense of what they are reading. They generally consist of the actual text in the document that contributes to content comprehension and the context of the document - building blocks of the document that enhance comprehension.

In machine learning, ‘feature selection’ is a way to choose which parts of a message are relevant for analysis, also known as ‘feature reduction’ or ‘feature extraction’. For email spam filtering, it is the selection of variable subsets that provide the most relevant features for the development of robust learning models for classification. Email spam classification systems deploy feature techniques for feature selection that process data, reduce the number of features, and remove irrelevant, redundant and noisy data from datasets. Feature selection in the simplest way is achieved using ‘Bag-of-Words’ (BoW), These represent the message as an unstructured set of tokens, sequence of characters that are separated by spaces and punctuation marks, which can be used to characterise the entire or part of the message. For example, the entire email message includes header and body while parts may be the body only, body and subject, the subject only or the header only. For greater sophistication, the occurrence of the same word in different parts of the message considers each event as different feature. This approach makes use of the structure of the email message; however, it does differentiate between the token present in the text of the body from the technical information in the header. Our Bayesian classifier (Chapter 3) uses this approach.

There are four types of feature selection methods - filter, wrapper, embedded and hybrid. The wrapper method starts with a set of all features, performs the classification using a learning algorithm and evaluates the performance of a feature subset. It performs cross validation to find the optimum feature set based on the gain in accuracy of the classification. This makes the wrapper method computationally expensive and does not work well with large datasets that contain large sets of features. The filter method is a correlation-based approach for the selection of features, is fast but is not capable of minimising the generalisation error. The embedded methods consist of algorithms that use feature selection and model fitting simultaneously using a sparsity regularizer and making the weights of some features zero. Another option to overcome the shortcomings of both, the filter and wrapper, methods is to use the fourth approach , hybrid, where some features of the wrapper method are used together with the filter method for feature selection (Guyon & Elisseeff, 2003).

Method of Semantic Feature Selection in DMLM:

To extract text features from the training data, the BoW approach has been applied to transforms data into numerical features that can be used for machine learning techniques. A BoW variant is useful where features are not just considered as binary but are regarded as weights based on the number of occurrences of the token in the message. The basic BoW can be enhanced by utilising alternative ways of feature selection such as stemming (removal of affixes), stopping (disregarding frequently occurring words) and lemmatisation (to reduce the word to its root form).

Our proposed model incorporates this enhanced BoW vector space model for selecting the best features; it uses term frequency to attach weights to the tokens to then store the features as a document term matrix and chi square test. ‘Term frequency’ is a feature extraction method where it transforms a text corpus into the vector of token counts and n-grams. This makes it easy to directly use this representation as features (signals) in machine learning tasks such as for email spam classification. This method has been selected as it provides a way by which collection of email messages are tokenised, bigrams have been selected to be used to build tokens and a vocabulary of known words is developed. Here, the number of times each token is present in an email is counted and this count is used to determine the weight of that token. Then, feature vectors are developed that are used to build the feature space and to be converted into a sparse matrix which stores the features identified from the email training corpora. This is followed by encoding of the new email message using the vocabulary developed from the training email dataset. For this study, this method is used to tokenise the spam and ham emails from the training datasets and develop the sparse matrix.

4.3The process of building the feature space is shown in Figure 4.5

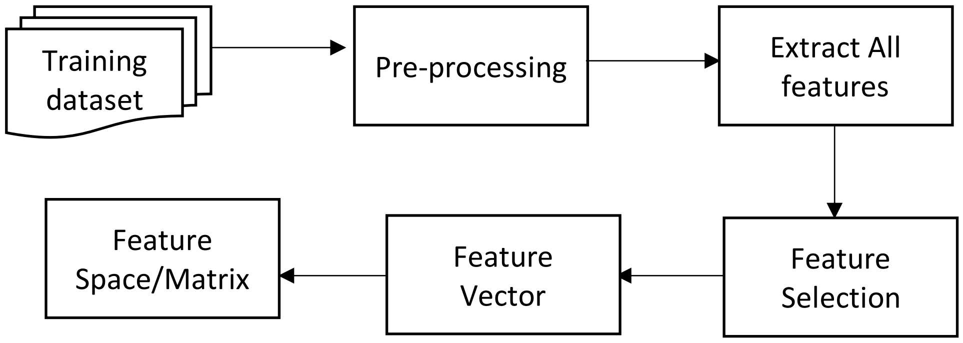


Figure 4.5: Building feature space using Semantic Features

This provides a sparse Document-Term Matrix M with a high number of columns. To apply machine learning algorithms to such a matrix, there is need to filter these columns. Different methods such as sparse latent special analysis, mutual information and chi-square test were tried for filtering, from which the chi-square test chosen due to its superior performance.

Chi square is another feature selection method for text categorisation which measures the lack of independence between the feature and output class (ham and spam) by computing the chi-squared statistics between each non-negative feature and the class. It assigns a value of zero if there is independence between the feature and the output class as those features are irrelevant for the task of classification. The score is used to select the n features with the highest value of the chi-squared statistic score. The selected features must contain a term count and be related to the output classes of ham or spam. This gives a feature space with n significant features that are used for training the classifier. The process flow model for feature selection is shown in Figure 4.6.

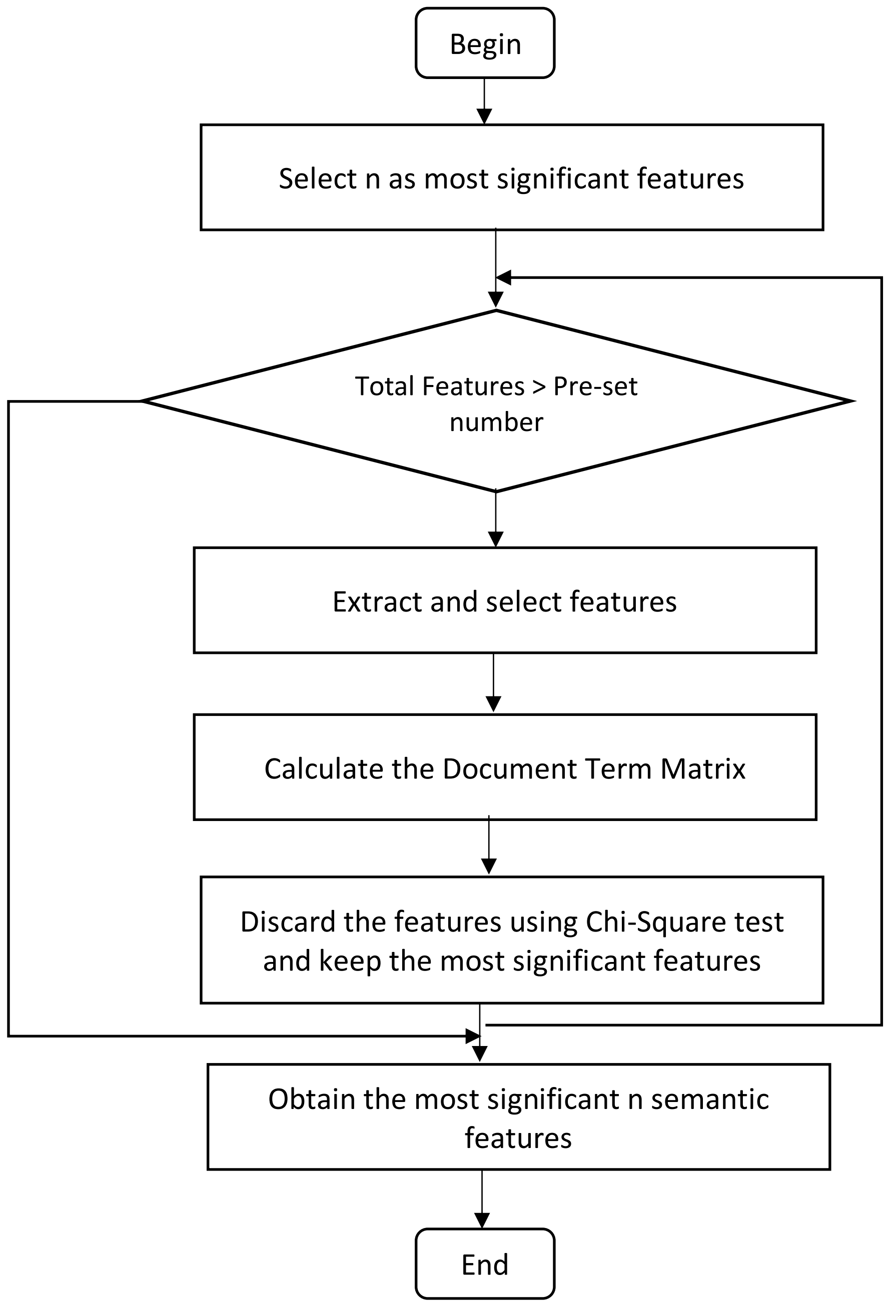

Begin
Select n as most significant features
Total Features > Pre-set number
Extract and select features
Calculate the Document Term Matrix
Discard the features using Chi-Square test and keep the most significant features
Obtain the most significant n semantic features
End

Figure 4.6: Process Flow model for Semantic Feature Selection

4.4 Syntactic Feature selection

It is easy to deceive learning techniques focused only on textual attributes in an email making it highly desirable to consider available additional features. This module analyses non-textual attributes and is of high importance as it selects features that provide additional, significant characteristics of ham and spam emails that impact the classification. Spam emails are highly diverse in topic and in nature with some having only content while others may be a mix of content and links or only links. They may also link directly to the web pages of spammers. Some spam messages contain only graphics (pictures, or animations, embedded or as an attachment) and some contain a mix of graphics and content. In many cases, graphics point to hidden links on the webpage of spammers. The difficulty is that the genres of spam emails match closely those of ham emails (letters, invitations, account summaries, notifications, order confirmations, offers, etc.) All these factors together produce high complexity and diversity which make the spam filtering process extremely challenging.

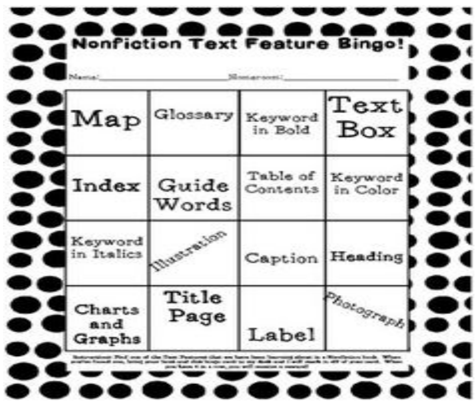
Spam filtering is further complicated by the complex nature of email spam data as spam topics range from pharmaceuticals, porn, shares and stock exchange to medicines and drugs, hardware and software, dating, online banking, healthcare, tax office refunds, account credential updates, online courses, workshop registrations, software updates, getting a degree, religious events, travel packages, holidays, gifts, job offers, threats, conference calls and more.

As discussed earlier in Chapter 3 (see learning email filter model), the learning capacity of a filter is highly dependent upon the email corpus used to train the model and is impacted by the shift in the nature of email spam. This section aims to analyse another facet of email spam to address this problem - anti-spam techniques - especially the filtering of technological endorsements through capability enhancement designed to prevent email spam from entrenching itself. In addition to applying another layer of anti-spam technique at the client level, training the spam classifier with relevant user data to act as control tool, concept drift, and identifying poison attacks can reduce the chances of FP and FN. The case that training the classifier with user data leads to reduced FP and FN has been proven in Chapter 3.

A possible solution, in this case, is to make the filters more efficient by adapting to user preferences, user behaviour, including syntactic structural features such as times of the day the spam email is received, and the dates/periods during which spam email is received. Such fine tuning could potentially further reduce the number of FP and FN. The introduction of non-textual features is also testified by the spam reports published by Kaspersky labs for Quarter 2, 2015 (Shcherbakova et al., 2015) which highlighted the variation in features identified in spam emails used by spammers to deceive the filtering solutions. These features are modified IP addresses, presence of upper case and lower-case letters, special characters, number symbols, mis-spelt words, and the number of links used to go to spam resources.

Method for syntactic feature selection:

Once the most significant semantic features have been selected, the syntactic features are extracted from the training dataset. This section focuses on listing which syntactic features are added to the feature set in order to distinguish between ham and spam emails. Syntactic features provide information about the document in terms of size, structure, illustrations, labels, subtitles, table of contents, glossary, maps, index, comparisons etc. as shown in Figure 4.7. In the case of emails, non-text information could include date and time of an email, subject field, hyperlinks, numeric digits, word count, use of special characters, etc.



Nonfiction Text Feature Bingo!

Name:__________ Homeroom:__________

Map	Glossary	Keyword in Bold	Text Box
Index	Guide Words	Table of Contents	Keyword in Color
Keyword in Italics	Illustration	Caption	Heading
Charts and Graphs	Title Page	Label	Photograph

[illegible]

Figure 4.7: Non-Fiction Text Features for a document

A series of small experiments were conducted to determine which syntactic features to include as described in the following sub-sections.

4.4.1 Experiment 3.1- Filtering based on time period

An interesting aspect is to focus on the time of the day email spam is received by the user to identify a correlation between the two although it is necessary to consider global time zone differences between the sender and receiver.

Method: A small experiment was conducted to gather data on timings of the spam emails received by a user and identify a pattern which can be used to enhance the classification model. In this experiment, a sample of email spam data of 463 emails was collected. The following business rules were developed for time-based filtering

- peak business hours-10am-4pm
- business hours-7am-6pm
- out of business hours-6-10pm
- late night hours-10 pm-7 am

An analysis was carried out to identify patterns in spam emails based on the time the email was received.

Results: The results of the analysis of email spam data indicated that 30 % of spam emails were received during peak business hours, 49% during business hours, 14% during out of business hours and 37% during late night hours. Hence, 51% spam emails arrived out of typical business hours of 7 am – 6 pm. This figure is high enough to determine that time-based filtering is an important feature in classifying an email as spam or ham.

4.4.2 Experiment 3.2- Concept drift – Temporal Filtering

Data changes with time. With this shift in the nature of data, data models also need to change to re-envision the conceptual model that may introduce instability in the system. Email spam filtering using machine learning algorithms is particularly challenging in the dynamic environment that email spam exists in as relearning is time consuming. Spammers continue to develop new methods to circumvent filters with learnt knowledge which leads to changes in data distribution and the concepts learnt by the filter changes over time. This causes 'concept drift' leading to expiry of concept patterns learnt from data samples that were up to date at the time. Furthermore, 'concept drift' brings about changes in the hidden concepts for which features are not shown explicitly but which are, nevertheless, needed for prediction; together, this leads to changes in the target concept (Koychev, 2006, 534). For example, sudden changes in the buying behaviour of customers. This causes samples to become outdated and no longer capable of correct prediction and classification. Effective machine learning should be able to adapt to these changes quickly; however, in the case of spam filtering, for concept shifts, the correct sample data for training are needed.

For spam emails, concept drift may be further divided into two types, sudden and gradual change, whereby email spam filtering is a victim of the former for which research for email spam filtering is underdeveloped. One of the few studies carried out in this area are from Delany, Cunningham, Tsymbal, and Coyle (2005) who developed a lazy learning case-based technique for dynamic learning. Their system uses kNN to retrieve k cases most relevant to the target case. No stop word removal or lemmatization is performed. A similarity retrieval algorithm Case Retrieval Nets (CRN) is used for flexible retrial of cases. However, this approach does not include any domain specific features. Fdez-Riverola, Iglesias, Díaz, Méndez, and Corchado (2007) present an instance-based reasoning model for detection of email spam also using a lazy learning algorithm. Here key terms and up to date email samples were the base data; however, classification accuracy is low, and the percentages of FP and FN are relatively high.

Sheu et al. (2017) proposed a method whereby the problem of concept drift for email spam filtering is addressed via a window-based approach. However, this method only considers the header section of the emails to reduce the computation cost which does not represent a comprehensive measure to address the issue of concept drift. Furthermore, none of the proposed techniques consider that concept drift affects individual users differently and no research at the user level has been presented so far.

Method: In order to verify the impact of concept drift on email spam, email spam attack data from 2013, 2015 and then from 2019 was collected for this experiment. A data analysis was conducted that focused on identifying the emails belonging to the identified ontological spread. The aim is to determine email spam attacks belonging to the identified ontologies in the said times.

Results: The following chart reports the results of the experiment showing the percentage of organizations affected by email spam attacks related to each topic.

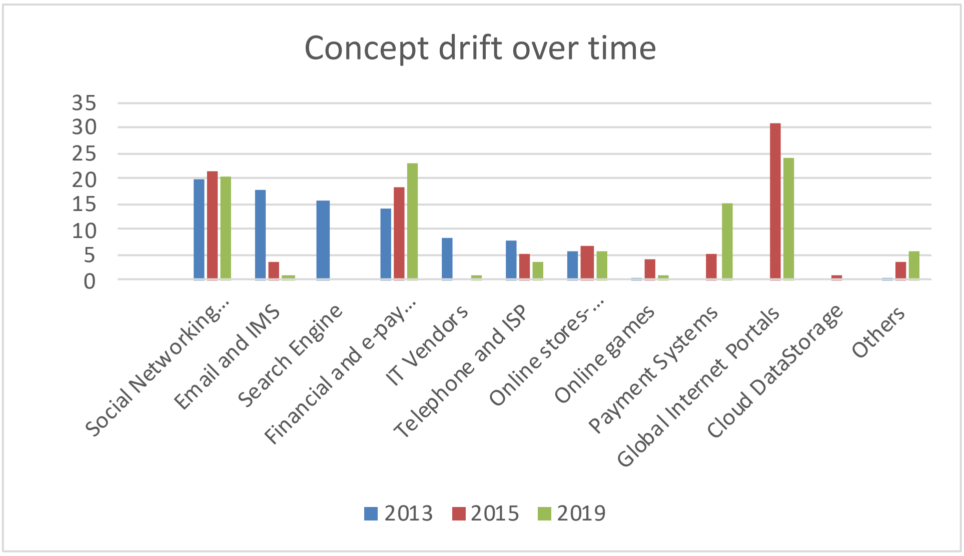


Figure 4.8: Distribution of data showing Concept drift from 2013 to 2019

It is evident from the statistics shown in Figure 4.8 that the topics of the email spam attack change with time. For example, the topic of Email and Internet Messaging System (IMS) was 17.61% in 2013 whereas it became negligible in 2019, which is similar in the case of Telephone and ISPs whereas attacks on Financial Organizations and Banks have gained momentum from 14.02% 2013 to 22.76% 2019. Therefore, incorporating filtering rules around such topics would increase the capability of the spam classification filters to identify spam. Hence, these topics were included in the DMLM syntactic features as 'abnormal' keywords.

4.4.4 Ontologies

Due to the fact that email spam is user dependent, there is a need for the filter to be user defined. The efficiency of user-based classification can be improved by using ontology-based concept definitions. Ontology plays an important role in defining the semantics of concepts and information for an adaptive and automated system like learning spam filter. It is defined as the representation of vocabulary of primitives, information, concepts, constraints and relationships. In database terms, these primitives can be entities, attributes or relationships between them. In email spam filtering, ontology can be used to define different categories to capture user preferences and build user profiles. Each of these categories may contain keywords, which are instances in an ontological language. Balakumar (2008) proposed an ontology for understanding the content of the email based on a white list and user defined categories. They performed user preference-based classification of spam and categorization of ham emails using a Bayesian approach. (Caruana et al., 2011) employed ontology-based concepts to minimize the impact of degradation of accuracy in email spam filtering when training data distribution using multiple SVM classifiers. Pham (2011) built an ontological knowledgebase to improve the performance of spam filtering at the server side using semantic information from the ontology. DMLM includes ontologies to improve the classification accuracy as a syntactic feature, ‘keywords’, where several categories and the corresponding keywords are defined. For experimental and testing purposes three spam ontologies were defined as “Sexual”, “Finance” and “Marketing”. For email spam classification, the model identifies spam keywords from spam emails; however, for ham categorization, this model requires user involvement. Hence, for minimal user involvement goal, this was ignored, and results were found to be promising.

4.4.5 Content and Word Obfuscation

Content obscuring is a common technique invented by spammers in response to rule-based filters where the rules are defined by users based on presence of certain words in headers or/and body of an email that are indications of spam.

A sample spam email that the Bayesian classifier can successfully classify is given in Table 4.1. Identification occurs through words such as ‘information’, ‘$2’, ‘1-800’ as spam words from which the email is classified as spam.

Table 4.1 Sample Spam email with Text Features

```
Subject: dental - optical plan


hello , work group local doctors dentists offer dental - optical
plan runs approximately $ 2 week individual $ 3 week entire
family . further details please call : 1-800 - 463-6021 toll
free please refer id code emjc56 p . s . call details before 13
, 1998 dental plan receive optical plan free ! thank .
```

Spammers continue to develop new ways to deceive the filters. The content of spam has evolved to contain more than just words such as links, numeric digits, special characters, etc. to substitute the text characters in an email to bypass the effectiveness of classification of email spam. These substitutions are generally not found in the ham emails and hence can be used as sign of low credibility of an email. Most of these features are syntactic such as grammatical errors as shown in the sample spam (Table 4.2 (a), numeric digits and are unlikely to be identified by the Bayesian classifier or any other text-based filtering mechanism.

Table 4.2 (a) Sample Spam email with Numeric digits

```
Subject: returned mail : host unknown ( name server : - - -
- - - . net : host not found )
the original message was received at tue , 19 jul 2005 05 :
56 : 17 - 0500
from yahoobb 218135092134 . bbtec . net [ 218 . 135 . 92 .
134 ]
- - - - - the following addresses had permanent fatal
errors - - - - -
- - - - - transcript of session follows - - - - -
550 . . . host unknown ( name server : - - - - - - - . net :
host not found )
```

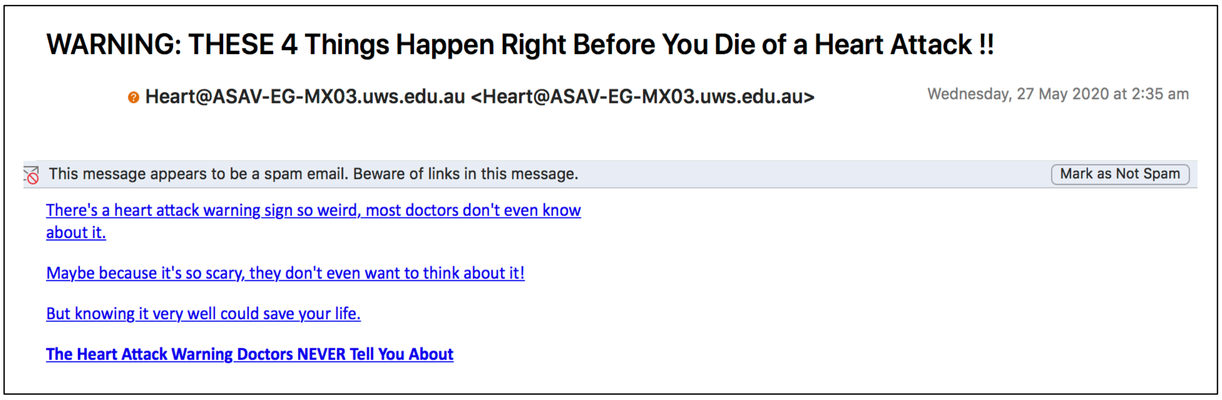
A deliberate use of grammatical errors in the words as shown in the email message body of sample email spam in Table 4.2 (b), this is designed in this manner to deceive text-based filters such as Bayesian classifier. Here, the aim of spammers is to introduce words that are not accounted for in the features used to calculate the spam score of an email message, as only the correct version of these known 'spam' words is identifiable by the content-based filters.

Table 4.2 (b) Sample Spam email with Mis-spelt words

```
Subject: new offrr
want to know ho liberalize w to save over 60 % on your me
northward dlcatlons ?
http : / / www . centr compassion alpan . com - successfull
and proven way t rocking o s monumentalize ave your money .
be splash st prlces .
high qua homophone iity .
w grundyism orldwide shlpplng .
total confidenti despoil aiity .
more than 200 appeasement popular medlcatlons
have a nice da upheave y !
```

Table 4.2 (c) shows the presence of links/URLs to obfuscate the textual part of the email message body. These words are not accounted for by the text-based filters. Another common spammer tactic is to obscure the words in the email content by including punctuation symbols, html tags and html comments in the middle of the words (Stern, 2008). Therefore, a mechanism to deal with such factors is required.

Table 4.2 (c) Sample Spam email with links as content


WARNING: THESE 4 Things Happen Right Before You Die of a Heart Attack !!

Heart@ASAV-EG-MX03.uws.edu.au <Heart@ASAV-EG-MX03.uws.edu.au> Wednesday, 27 May 2020 at 2:35 am

This message appears to be a spam email. Beware of links in this message.

There's a heart attack warning sign so weird, most doctors don't even know about it.

Maybe because it's so scary, they don't even want to think about it!

But knowing it very well could save your life.

The Heart Attack Warning Doctors NEVER Tell You About

Therefore, as suggested by Bajaj & Pieprzyk (2013), to improve the performance of the Bayesian classifier, several non-textual syntactic features identified in this research (shown in Table 4.3) were included in DMLM. To extract all important syntactic features for inclusion in the DMLM, an analysis of spam datasets was carried out to identify characteristics that are distinctly different from legitimate emails to find an optimum list of attributes as potential features.

Table 4.3: Syntactic Features

0: email header and body lengths,
1: number of abnormal symbols,
2: number of numeric characters,
3: number of punctuation symbols,
4: number of links symbols,
5: 'number of keywords',
6: number of keyword ‘unsubscribe’
7: 'send time in 8:00:00 - 18:59:59,
8: length of subject field,
9: mis-spelt word count
10: similar to abnormal words
11: maximum run length of capitals
12: average run length of capitals

To select the relevant features, a manual cross validation was carried out with the classifier. Once the syntactic features were selected, parsing of the text data was performed with removal of encoded text and html tags to include those words for semantic features. A relevant feature is one that enables the classifier to improve its performance. To determine the best performing features, the F1 score was used as a metric.

The F1 score, also known as the F-score is the measure of accuracy of a test. It is the balance between precision and recall. It is an appropriate measure to use when the actual negatives are larger than the actual positives. In our case, the number of ham emails in the samples was larger than that of the spam samples; hence, the F1 score was used for selecting features as it enables further classification. The F1 score is shows a satisfactory result when it is high and close to 1, perfect when it is 1 and at 0, the model is at its worst performance. Good F1 scores mean there are low FP and low FN, indicating that detection is being carried out correctly and was not polluted by false alarms. To determine the appropriate features, the classifier's F1 score without feature was recorded as *F_without* and the F1-score with feature recorded as *F_with*, then the feature is the "right" feature if *F_with* > *F_without* with statistical significance.

All of the features selected were put together into a feature set as shown in Table 4.1 that shows 13 syntactic features used in DMLM. As discussed earlier, there is limited research into the use of syntactic features for email spam detection at the client side. Gargiulo (2009) developed a personal anti-spam system at the server level where an architecture based on semantic and syntactic features from text and image spam was used for classification to overcome the problems inherent in the state-of-the-art spam classification filtering systems. However, the syntactic features included in that model were limited. Hence, this research is a novel contribution in this field.

4.5 Model Generation method

Once feature extraction and selection are complete, these features are used as attributes for training and classification via the model which is utilised to detect the similarity between ham/spam emails in a new email. It is developed by using the selected significant feature set and training samples. In this section, the process of the model generation is explained.

The value of the most significant features from all ham and spam training samples, the training samples detection threshold values and the machine learning algorithm/s are the basic components needed to generate the model. A set of training samples from ham and spam $X,Y = \{x_1,x_2,\dots x_i\}$ $\{y_1, y_2,\dots y_j\}$, given at the feature selection stage, and the most significant features $F = [f(h_1), f(s_1), f(s_2), f(h_2),\dots,f(s_n),f(h_m)]$ where $f(h_1), f(s_1), f(s_2), f(h_2),\dots,f(s_n),f(h_m)]$ are the combined features to detect spam and ham in emails used to train the model.

The model is generated using an ensemble of linear combination of 200 CART trees which is a boosting decision tree ensemble. An ensemble is a collection of combined outcomes to give a final outcome as prediction. The purpose of ensemble learning is to combine the predictions from several classification models with an aim to minimise the generalization error and the variance in predictions made by different models. The main cause of variations is noise, variance and bias.

Ensembles can be formed through the application of different methods for combining predictions. Research has verified that ensembles perform better than individual classification models (Chinavle, Kolari, Oates, & Finin, 2009; W. Wang, 2010). Ensembles can be formed using one of three techniques - bagging, boosting or stacking. Bagging develops multiple independent models and combines them using an averaging technique. Most commonly, independent models are built based on sub samples of data from training data; models developed from each sub sample would be little different from each other. Predictions from each of the models is combined using normal/weighted averaging or majority voting, such as decision trees or random forest algorithms. Stacking is an ensemble which takes the output from collections of other models as input and then produces predictions. The main aim here is to reduce overfitting and improve accuracy. The third technique, boosting is an ensemble technique that does not make independent predictions but rather sequential ones. It converts weak learners to strong learners by passing the learning from the errors made by previous predictors to the subsequent predictors. Due to this technique of learning from previous predictors, the iterations/time taken to reach the actual correct predictions is low although it still reduces bias and improves accuracy. Typical examples of the latter technique are Classification and Regression Trees (CART), gradient boosting, XGBoost or AdaBoost.

The model chosen is a boosting ensemble of classification and regression decision trees. Classification and Regression Trees (CART) algorithm, based on work by Breiman et al (1984), is a supervised learning technique for prediction, and classification.

The model initially takes the feature set F defined from the training samples for the ham and spam classes as described in the previous section. To this, it adds the two output classes ham and spam as starting point and builds decision trees based on the attributes of the feature set. It then constructs the rules to make progressive hierarchical decisions about the new email as ham or spam.

The algorithm identifies the class attributes from the training data and constructs decision trees based on those identified attributes, which requires compilation of the attribute list and definition of the number of output classes. Subsequently, CART constructs the rules via the decision trees from the training data with the assigned classes. These decision trees are then used for classification of new data. It is a binary decision tree developed by repeatedly splitting nodes into their child nodes representing the entire training data and refers to the two types of trees - classification trees and regression trees (Marsland, 2014). Based on existing features, the decision tree makes progressive, hierarchical decisions about the outcome. The algorithm is structured as a series of questions that result in a tree-like structure based on further questions about answers to the previous set of questions. The aim of this process is the development of a prediction model. The three important components of CART are: the splitting and stopping criteria and prediction.

The aim of tree growing is to identify the univariate split of every node so that impurities in child nodes are reduced as much as possible. The splitting criteria represented by $\Delta_{(s,n)}$ are maximised by the split s at each node n. Ginni's coefficient is used as the splitting criteria which is an index of impurity for every node, with the aim of progressively minimising it until the node is pure. Ginni's impurity index for sample data D with x classes is defined as

$$Ginni(nD) = \sum_{i=1}^{x} P_x^2$$

Calculating Gini's coefficient is the most time-consuming part of the CART algorithm and is used to simplify the model rather than information gain ratio to identity the features (Zhu, 2018). The quality of the features is determined by the lower value of the Ginni coefficient, which means this index is calculated for all features and the ones with lower values are selected.

The stopping rule defines at which point the node cannot be split any further. The following cases aid in stopping the split: when all cases in a node have identical values for each feature or its dependent variable, when the node size is smaller than the user defined minimum node size, or when the tree depth has reached the user defined maximum depth. The quality of the learnt tree can be improved by pruning which involves removing those parts of the tree that do not contribute correctly to classification. Since decision tress are prone to overfitting, pruning can lower this likelihood.

The next component involves prediction, given that a novel input CART traverses the constructed decision binary tree to make predictions. In CART, the leaf only contains the decision values which sets it apart from other decision trees and allows rich interpretations for optimisation.

The algorithm for generating the model for classification is based on a machine learning process as described in the previous section:

- Read email from training set samples
- Perform pre-processing
- Develop Semantic features using modified BoW model
- Select n significant features and develop a feature document term matrix
- Develop syntactic features
- Combine all features
- Build the classification model
- Train the model using training samples and store for later use
- Perform classification using the stored trained model

Classification is done using the generated model.

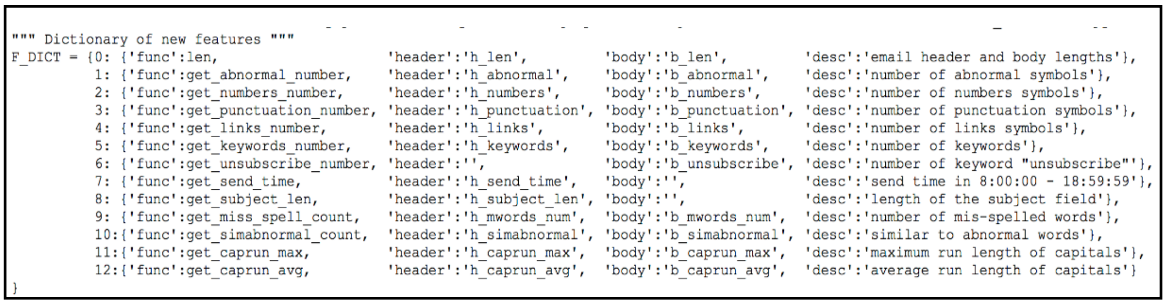
4.6 Classification Results and discussion

In this section, the results of the experiments are presented and analysed. The evaluation of the feature selection approach is carried out using several datasets, each divided into two subsets in a 70:30 ratio, where 70% is used for training and 30% for evaluation. The training subset consists of sample ham and spam email messages. The BoW approach using a Term Frequency method is used on each dataset of training samples to extract significant features for training the model to be used for the classification process. Term frequency is useful for eliminating noisy tokens since they present insignificant information for classification; hence, their removal has little influence on the overall performance. The computation complexity of term frequency rises linearly with the increase in the number of training messages. Therefore, the method is reasonable inexpensive in terms of computation cost.

The experiments are performed in three steps:

- The first step involves iterative feature selection. A heuristic search method using a BoW approach is used to select the number of tokens and n-grams as a starting point and then the Term Frequency method is used to add weights to tokens to create a sparse matrix. THE Chi-square test is used subsequently to obtain the optimal features set.
- The second step involves selecting the syntactic features from the training samples as shown in Table 4.4.
- In the third step of the experiment, the selected feature set is used to develop the model trained on the training samples using an optimal feature set, which is then used to classify new, incoming emails as ham or spam. In this case, the test set was used as new emails for experimental purposes to measure the performance of the DMLM.

Table 4.4: Syntactic features list during the Feature selection Step



```
""" Dictionary of new features """
F_DICT = {0: {'func':len,                    'header':'h_len',         'body':'b_len',          'desc':'email header and body lengths'},
          1: {'func':get_abnormal_number,    'header':'h_abnormal',    'body':'b_abnormal',     'desc':'number of abnormal symbols'},
          2: {'func':get_numbers_number,     'header':'h_numbers',     'body':'b_numbers',      'desc':'number of numbers symbols'},
          3: {'func':get_punctuation_number, 'header':'h_punctuation', 'body':'b_punctuation',  'desc':'number of punctuation symbols'},
          4: {'func':get_links_number,       'header':'h_links',       'body':'b_links',        'desc':'number of links symbols'},
          5: {'func':get_keywords_number,    'header':'h_keywords',    'body':'b_keywords',     'desc':'number of keywords'},
          6: {'func':get_unsubscribe_number, 'header':'',              'body':'b_unsubscribe',  'desc':'number of keyword "unsubscribe"'},
          7: {'func':get_send_time,          'header':'h_send_time',   'body':'',               'desc':'send time in 8:00:00 - 18:59:59'},
          8: {'func':get_subject_len,        'header':'h_subject_len', 'body':'',               'desc':'length of the subject field'},
          9: {'func':get_miss_spell_count,   'header':'h_mwords_num',  'body':'b_mwords_num',   'desc':'number of mis-spelled words'},
          10:{'func':get_simabnormal_count,  'header':'h_simabnormal', 'body':'b_simabnormal',  'desc':'similar to abnormal words'},
          11:{'func':get_caprun_max,         'header':'h_caprun_max',  'body':'b_caprun_max',   'desc':'maximum run length of capitals'},
          12:{'func':get_caprun_avg,         'header':'h_caprun_avg',  'body':'b_caprun_avg',   'desc':'average run length of capitals'}
}
```

4.6.1 Experiment Results

To verify the performance of the proposed algorithm for the DMLM, the training and evaluation was carried out using five datasets listed in Table 4.2., executed with each of the datasets. The description for all datasets except CSDMC 2010 is given in Chapter 3.

CSDMC2010: This dataset resulted from a data mining competition held under ICONIP 2010. Available on GitHub, the dataset size is 8619 which is split into 4,327 messages for training of which 2,949 are legitimate and 1378 are spam email messages. 4,292 unlabelled messages are available to be used for testing. The last update on the dataset was done in 2014. In this research all labelled samples, 4,327, are used for training and testing to enable the performance measurement of the models.

In Table 4.5, the column Number-Training shows the training samples for each dataset, i.e., the number of ham email as h and spam email as s messages in the training samples. All datasets have a balance of training samples belonging to each of the output classes. Hence, the issue of imbalance in the datasets is addressed since the number of samples of each class type i.e., ham and spam are well represented and not out of proportion.

The training samples were used to identify first, the most important semantic features and then the syntactic features.

Table 4.5: Number of Training and Testing Emails for each Dataset used

Dataset Name	Total Emails	Number - Training	Number – Testing
PU1	1,100	770(426h+344s)	330
Ling-spam	2,893	2,025(1727h+342s)	868
TREC07	4155	2355(1095h+1260s)	1800
CSDMC2010	4327	3052(2086h+966s)	1275
Enron1	5172	3666(2600h+1066s)	1506
Enron2	5857	4044(3006h+1038s)	1813
Enron3	5512	3868(2840h+1028s)	1644
Enron4	6000	4214(1060h+3154s)	1786
Enron5	5175	3632(1072h+2560s)	1543
Enron6	6000	4173(1049h+3124s)	1827

To select the important features, experiments were conducted to choose n-gram for iterative feature selection. The candidate n-grams were unigrams and bigrams. Out of 10 datasets, 3 datasets were randomly chosen, an initial feature set was created using unigrams and bigrams. The total number of features identified using BoW, time taken to identify the features for each of bigrams for the Enron 6 dataset with sample size of 3868 is shown in Table 4.6 below.

Table 4.6: Semantic Feature Extraction using n-grams

n-grams	Features extracted	Time taken (seconds)	Processing Speed MB/s	Training F1 score)	Test F1 score
Unigrams	46206	1.846	5.483	98.9	98.4
Bigrams	470345	7.939	1.275	99.4	98.6

The term frequencies of the features were calculated and the features with smaller value of term frequency were neglected. It was assumed that these features had little influence on the classification process and performance. The weights were attached to the remaining features. As a starting point, 1000 features were selected as significant features and the chi-square test was used to identify the most significant features to be used for training and classification of the model. Evaluation of the model was carried out using first unigrams and then bigrams with results of the classification reported in Figure 4.9 where 'without new factors' refers to unigrams and 'with new factors' to bigrams.

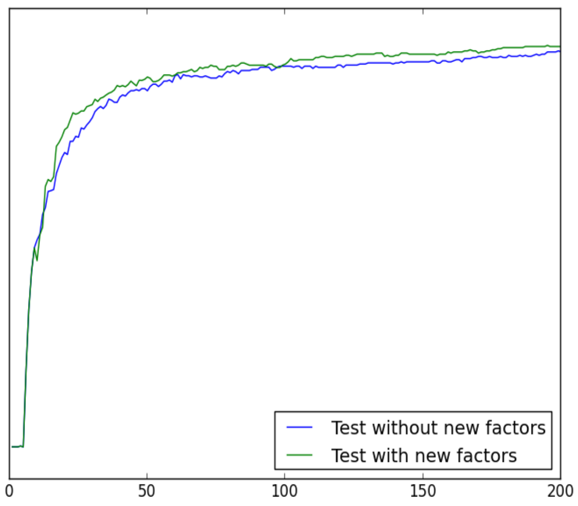


Figure 4.9: The classification accuracy using unigrams and bigrams as features.

The blue line represents accuracy for unigrams and the green line for bigrams. The F1 score for training and testing of unigrams and bigrams are shown in Table 4.6. It can be observed from these figures that bigrams perform better than unigrams. To validate the results, several experiments with additional datasets were conducted. It was observed that optimal results were achieved for bigrams for our iterative feature selection using semantic and syntactic features. All the selected features were integrated into one feature vector. Hence, bigrams were used for semantic feature selection, model training and classification. For each dataset, the number of features extracted, training time taken, and evaluation time is shown in Table 4.7 below:

Table 4.7: Semantic Feature Selection, Training and Test times

Dataset	Number of Sementic eatures (bigrams)	Feature extraction duratiom (seconds)	Significant Feature selection duratiom (seconds)	Speed (MB/sec)	DMLM Train time	DMLM Test Time
CSDMC2010	728598	15	0.91	1.27	53	17
ENRON1	247949	3.84	0.28	1.13	58	24
ENRON2	318353	5.8	0.32	1.1	64	21
ENRON3	470345	9.1	0.43	1.1	69	25
ENRON4	329480	4.7	0.26	1	72	22
ENRON5	272177	4.5	0.27	1.1	72	28
ENRON6	368384	5.9	0.33	1.05	66	25
LINGSPAM	419467	6.2	0.31	0.9	65	19
PU1	152526	2.6	0.14	0.75	13	15
TREC07	599843	18.2	0.96	1.03	72	24

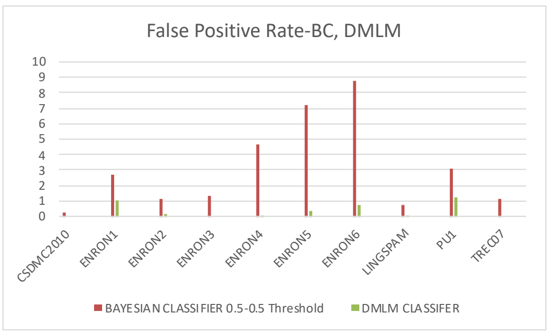
During the evaluation stage, the performance of selected features and models is evaluated on the test data subset containing ham and spam emails for each of the datasets in Table 4.5. The results are presented for the FP and FN rate and accuracy. For each of the datasets, the results are shown in the tables to follow.

Table 4.8: a) False positive and b) false negative rate for all 10 datasets for Bayesian classifier and DMLM

DATASET	METRIC	BAYESIAN CLASSIFIER 0.15-0.9 Threshold	BAYESIAN CLASSIFIER 0.5-0.5 Threshold	DMLM CLASSIFER
CSDMC2010	FP RATE	0	0.23	0
ENRON1	FP RATE	1.3	2.7	1
ENRON2	FP RATE	0	1.1	0.2
ENRON3	FP RATE	0.1	1.36	0
ENRON4	FP RATE	0.1	4.7	0.1
ENRON5	FP RATE	0.1	7.2	0.4
ENRON6	FP RATE	0.2	8.8	0.7
lingspam	FP RATE	0	0.72	0.1
PU1	FP RATE	0.9	3.09	1.2
TREC07	FP RATE	0	1.1	0

DATASET	METRIC	BAYESIAN CLASSIFIER 0.15-0.9 Threshold	BAYESIAN CLASSIFIER 0.5-0.5 Threshold	DMLM CLASSIFER
CSDMC2010	FN RATE	0	1.69	0.7
ENRON1	FN RATE	0	3.9	1.4
ENRON2	FN RATE	0.1	5.02	1.4
ENRON3	FN RATE	0.1	5.7	1.3
ENRON4	FN RATE	0.3	0	0.3
ENRON5	FN RATE	0.2	0.89	0.2
ENRON6	FN RATE	0.3	0.07	0.3
LINGSPAM	FN RATE	0	9.3	1.3
PU1	FN RATE	0	2.16	1.2
TREC07	FN RATE	2.9	0	2.9

Table 4.8 a) shows the comparison of the FP rate and Table 4.8 b) for the FN rate for the Bayesian classifier at a threshold value of 0.9 for spam cut off and 0.15 for ham cut off. It also shows the threshold values of 0.5 for both spam and ham cut off against the FP and FN rates for the proposed dynamic multi-layer model with one classification machine learning algorithm using CART. Here two threshold values for the Bayesian classifier are included for comparison since the performance of the classifier at 0.5 threshold values is comparable to the DMLM. At threshold values of 0.9-0.19, the classifier generates a substantial number of ‘grey’ verdicts which are left unclassified.



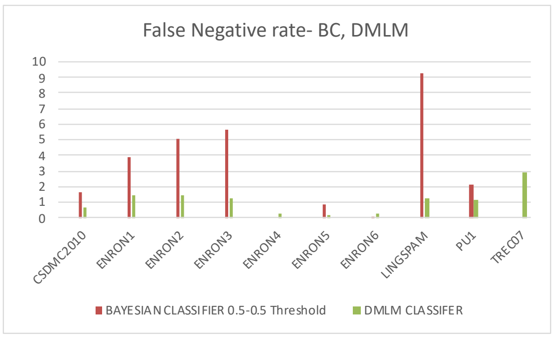


Figure 4.10: Graph showing a) False positive rate and b) False Negative for all 10 datasets for Bayesian classifier and DMLM

Graphs in Figures 4.10 a) and 4.10 b) show the performance of the Bayesian classifier and the proposed DMLM model on the evaluation test datasets.

Table 4.9: Accuracy for all 10 datasets for Bayesian classifier and DMLM

DATASET	METRIC	BAYESIAN CLASSIFIER 0.15-0.9 Threshold	BAYESIAN CLASSIFIER 0.5-0.5 Threshold	DMLM CLASSIFER
CSDMC2010	ACCURACY	100	98.08	99.3
ENRON1	ACCURACY	98.7	93.4	97.6
ENRON2	ACCURACY	99.9	93.88	98.4
ENRON3	ACCURACY	99.8	92.94	98.7
ENRON4	ACCURACY	99.6	95.3	99.6
ENRON5	ACCURACY	99.7	91.91	99.4
ENRON6	ACCURACY	99.5	91.13	99
LINGSPAM	ACCURACY	100	89.98	98.6
PU1	ACCURACY	99.1	94.75	97.6
TREC07	ACCURACY	97.1	98.9	97.1

Table 4.9 presents a comparison of the accuracy measure for the classifiers. Accuracy, a metric for evaluating classification models, is also known as Accuracy Classification Score (ACS). Accuracy is a measure of closeness of predicted value and actual value which is determined by the total number of predicted outcomes against actual outcomes. ACS measures the number of correct predictions as a proportion of the total predictions made for the test data. Performance of the classifiers in question - Bayesian classifier with DMLM is shown in Figure 4.11. The analysis of the results is discussed in the next section.

4.6.2 Analysis of Results of Experiment

The performance of two-layer model for email spam detection has been evaluated. The detection rate, FP rate and FN rate are presented in Tables 4.5 and 4.6. For different datasets, the proposed model was able to detect 97.1-99.6% spam email messages correctly with a low FP rate of maximum 1.2% and as low as zero; the FN rate ranged between 0.2-2.9%. The FP rate for the proposed model improved significantly as compared to the Bayesian classifier - visualized through Figure 4.10 a).

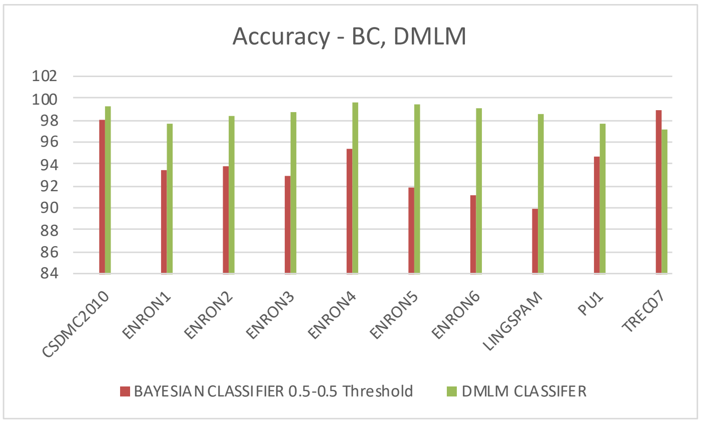


Figure 4.11: Graph showing accuracy for all 10 datasets for Bayesian classifier and DMLM

From Figure 4.11, it is clear that the DMLM outperforms the Bayesian classifier in accuracy and Figure 4.10 confirms that the proposed model performs better in terms of FP and FN rates. With this new classification model acting as another layer of filtering in addition to the proposed Bayesian classifier as explained in Chapter 3, it is evident that the DMLM performance is efficient. For four out of ten datasets, the proposed model performed more than 99% of correct predictions for email spam detection. From Table 4.6, it shows that the performance of the model in terms of computation complexity and time has been proven to be efficient.

Since there is no evidence found for any other multi-layer model to detect and filter email spam at the client side, a direct comparison with similar models cannot be presented in this section. However, a comparison of the performance with other machine learning based models can be found in Table 4.10 where data demonstrate that the DMLM outperforms other classifiers in accuracy and coverage of features including text (semantic) and structural (non-text) features. For Gomez & Moens (2010) and S. K. Trivedi & Dey (2013) results are acceptable in terms of accuracy although their feature selection depends on text features from only the body and header or just the body.

Table 4.10: Comparison of performance of DMLM with other machine learning classifiers

Paper Reference	**Machine Learning method(s) Used**	**Dataset Used**	**Accuracy**	**Features**
1 (Sheu et al., 2017)	C4.5	TREC07	95.5	non text from header only
2 (M. Sharma & Kaur, 2015)	AdaBoost, Random Forest	Spam Assassin	93.6, 93.3	Text features from body only
3 (Shrawan Kumar Trivedi & Dey, 2014)	Genetic Algorithm, Genetic Programming	Spam Assassin	93.2, 97.8	Text features from header and body
4 (Khater, 2012)	Random Forest	CSDMC2010	95.8	Text features from header only
5 (Karthika Renuka, Hamsapriya, Raja Chakkaravarthi, & Lakshmi Surya, 2011)	Multilayer Perceptron, J48, Naïve Bayesian classifier	UCI repository	93, 92, 89	Text (5 features) + Non-text-(3 features) from whole email
6 (Xiao et al., 2010)	Decision Tree	Chinese emails	96.5	Non-Text from header only
7 (Gomez & Moens, 2010)	C4.5	Ling-spam, TREC07	98, 99	Text features from body only
DMLM	CART	Enron, CSDMS2010, TRCE07, PU1, Ling-spam	99, 99.3, 97.1, 97.6, 98.6	Text (1000 features) + Non-text-(26 features) from whole email

The FN rate represents a misclassification of spam emails as ham. From Figure 4.12 a), it can be observed that the FN rate has improved with the addition of the DMLM; however, there is scope for reducing the FN rate further and, thus, reduce email spam in users' inbox. To achieve this an optimum FN rate needs to be determined that keeps the detection rate high while reducing the FN rate to an acceptable level. To further improve the performance of the model, it is proposed to use an ensemble of bagging classifiers as spam detection system called dynamic multi-layer ensemble model (DMLEM). The detailed description of the model is given in the next section.

4.7 Experiment 4-DMLEM - Dynamic Multi-Layer Ensemble Model

The DMLM discussed in Section 4.6 has been proven efficient in increasing the spam detection rate while reducing the computation time and complexity. However, the results presented in Section 4.6 indicate that there is room to improve the DMLM performance due to its detection rate for some datasets such as TREC07 and PU1 being lower than others. The FN rate for the DMLM was also noted to be somewhat higher than the Bayesian classifier at its best performance of thresholds 0.9-.015. High FN rates mean more email spam.

To further enhance the performance of the DMLM, this section proposes to build a bagging ensemble of machine learning classifiers. The rationale for including a bagging ensemble instead of using a single classifier is twofold: (1) to improve the classification performance and (2) to collate different outputs produced under varying conditions by different classification algorithms into a single output. As the nature of email spam content continues to change, bias is introduced when using a single classifier which impacts detection performance and increases the FP and negative alarm rates. Ensembles have proven to outperform single classifiers (Beigy, 2012; Rayana & Akoglu, 2016). Hence, it is intended to replace the CART classifier with the bagging ensemble CLKS to change the DMLM into a Dynamic Multi-Layer Ensemble Model (DMLEM) to perform classification of emails into spam and ham. In this section, experiments are carried out using the proposed CLKS ensemble to determine if the DMLEM outperforms the DMLM. The aim is to improve the performance of FN rates while maintaining low FP rates.

Dynamic Multi-Layer-Ensemble Model Description

While the proposed DMLM multi-layer model enhanced the classification performance, a hybrid classifier is based on supervised alternative machine learning techniques applied to the selected features using the semantic and syntactic components of an email. This classifier carries out supervised learning, extracts text as well as non-text features from training data and applies that learning to detect and classify new email documents.

The proposed ensemble model modifies Layer 2 of the DMLM to increase the FN rate and accuracy of classification - the architecture of the DMLEM is shown in Figure 4.12. It is of high importance that this model achieves a high level of performance for correct classification of spam and ham (or least FN) while keeping accuracy and FP rate constant at the rates achieved in Section 4.6. In the DMLEM, the methodology for feature selection (to extract significant features) is preserved as for the DMLM initially (as explained in earlier sections). The feature selection methodology may be modified based upon the results of the experiments with the DMLEM model, in case the model deteriorates in performance as compared to the DMLM. The following sub-sections explain the selection of machine learning algorithms and how they were combined to build the multi-layer ensemble model.

This model includes the following supervised machine learning algorithms: CART from DMLM, Support Vector Machine, *k* nearest neighbour and Logistic regression for classification (Chao & Yiming, 2007; Drucker et al., 1999). For sparse data or matrices commonly used machine learning techniques are SVM, Naive Bayes, XGBoost and Logistic Regression, applied with satisfactory results and speed of execution.

These methods utilize the decision boundaries that they identify from the training data and apply them for classification. The detailed description of how these methods are applied has been provided further. Each of these methods individually classifies the spam and grey emails categorized by the Bayesian Classifier into spam and ham. The multi-layer model further uses rules on these classifications to predict an email as spam or ham.

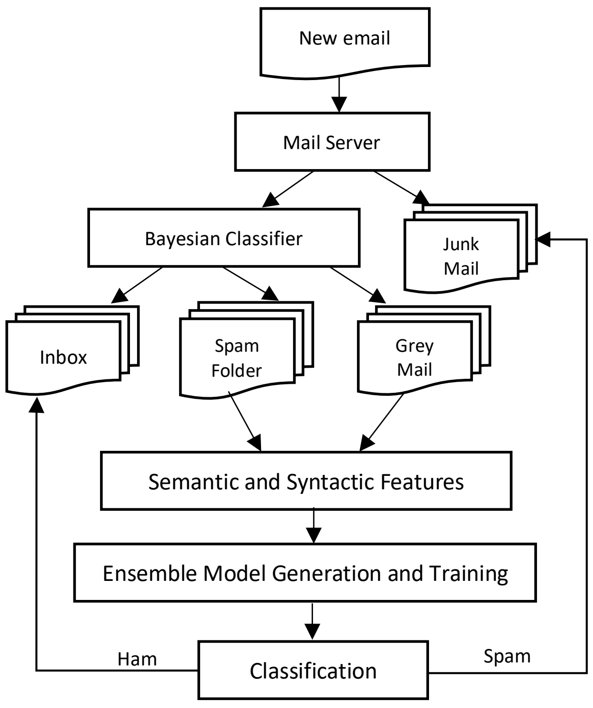


Figure 4.12: Architecture of DMLEM

A description and justification of these algorithms is given below:

A Support Vector Machine (SVM) is a non-parametric supervised machine learning algorithm developed by Vapnik (1998) with a strong theoretical foundation based on *Statistical learning Theory*. SVM, highly efficient on small datasets, is a non-probabilistic algorithm that analyses data for classification, regression and outliers (Cristianini, Nello, & Shawe-Taylor, 2001). It provides the probability distribution of a dataset given a finite set of parameters. It requires labelled data for training itself (Drucker et al., 1999). In this case the labelled data consists of the training email messages identified with the label spam or ham.

The SVM uses support vectors - a small group of data points identified from a collection of training points. It aims to solve the optimization problem by finding boundaries with maximum margin and maximum distance between the data points. The goal is to find a hyperplane in a multi-dimensional space with n features for distinct data point classification. A hyperplane is a decision boundary that segregates the data points into n classes. Many hyperplanes of dimension up to n-1 are possible. For binary classification, the objective is to find a hyperplane with maximum distance between the groups of data points with each group belonging to one of the two classes. Reinforcement of data points as belonging to a particular class occurs with increased margin distance from the hyperplane which leads to increased confidence for new data point classification in future.

The position and orientation of the hyperplane is influenced by the data points closest to it. These data points, called support vectors, facilitate building of the SVM as shown in Figure 4.13. Support vectors are coordinates of an individual data point. Hence, classification is performed by finding that hyperplane to differentiate the two sets of data points from common features.

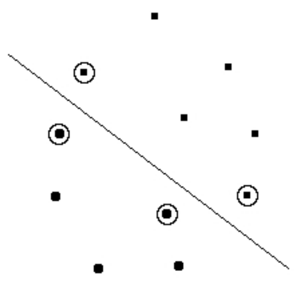


Figure 4.13: A One-Dimensional Hyperplane Separating the Data Points

Depending on the number of features, the hyperplane can be one-dimensional with 2 features or two-dimensional with 3 features. Although commonly used for linear classification, the SVM can also be applied to non-linear classification with kernel trick to map inputs to high dimensional feature spaces.

Assuming we have a hyperplane of the form $wx + b = 0$ with an aim to divide the training data x into two classes, where the vectors w and b are determined using x and are the same size as x. Ideally, there should be two hyperplanes with maximum possible distance and no data points between them to be represented as $wx + b = +1$ and $wx + b = -1$

In Figure 4.13, the aim is to find a, b, c such that $ax + by \geq c$ for all cube points and $ax + by < c$ for all dot points. A significant number of solutions are possible for finding a, b and c; however, it is important to note the points that just touch the boundary of the hyperplane and influence optimal values - circles in this case. There are four data points: two above the hyperplane, say $v1, v2$ and two below, say $v3, v4$.

To find the optimum solution, Figure 4.14 shows the two hyperplanes H_1 and H_2 represented as dashed lines defined as shown such that

$$wx + b \geq +1 \text{ when } y = +1$$

and

$$wx + b < -1 \text{ when } y = -1$$

the points on these two planes are the tips of support vectors – and + as shown.

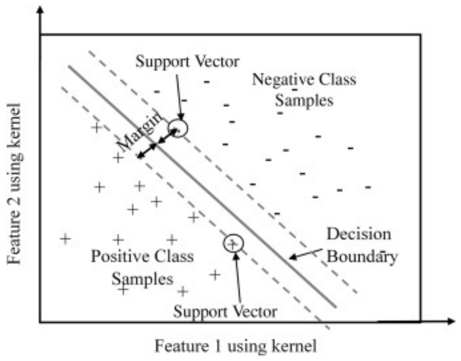


Figure 4.14: Support Vector Machine showing one dimensional hyperplane

The hyperplane (solid line) shown in Figure 4.14 is the median between H_1 and H_2 where $wx + b = 0$. The purpose of defining these two hyperplanes is to maximise the distance between the data points (Kala, 2016). The SVM classifies by finding a hyperplane that separates labelled data while maximizing the margin. Margin is the gap between the two categories of labelled data. Choosing a hyperplane that maximizes margin gives us a natural boundary for separating the data points. In total, there are 1026 features, each of which is a numerical value. So, each email is represented by a vector of dimension 1026. All training emails can be considered as points in a 1026-dimensional space with the spams and the hams as data points. In such a space separating data points on either side a plane would be called the classifier boundary, also known as a hyperplane.

Since most problems are non-linear when the data points cannot be separated by a plane, there is a need to make them linearly separable. Hence, a transformation function $\emptyset$ is required for mapping to high dimensional space. Here, the kernel trick function is used for transforming data points into a higher dimensional space so that they become linearly separably. To improve the performance of the SVM, Radial Basis Function (RBF), a kernel function, was used which can transform low dimensional spaces to higher dimensions for non-linear classifiers.

The SVM is typically selected for two group classification problems such as text classification. It produces accurate outcomes with lower cost in terms of computational power. It has been used in its original form and with kernel tricks such as distance-based kernel, linear kernel and Gaussian kernel to improve performance and also as part of an ensemble model (Amari & Bouguila, 2010;

Krawczyk, Minku, Gama, Stefanowski & Wolnik, 2017; Singh, Pamula, & Shekhar, 2018; Mellampati, Shekar, & Ravikanth, 2019).

K-Nearest Neighbors (kNN) is a non-parametric lazy learning supervised learning algorithm applied for statistical estimation since the 1970's. It is a feature similarity density estimation approach, as the name suggest, uses the labels of k-nearest training data points - also known as observation or collection - in the feature space to identify the label of the test data point under consideration. These k training data points form an input set which is used by the algorithm to determine the label of the test data point under consideration, based on its similarity measure to those labelled k data points. Here, k is the number of nearest neighbours. The output depends on whether the algorithm is used for classification or regression. Classification has a discrete value as output whereas regression shows a real number. It is called 'lazy' as it makes no generalisations, which means it does not generate models from training data, all of which is used within the test space. Hence, only the function approximation is performed locally, and all computation takes place at the time of classification. While this makes training faster, the testing becomes cumbersome and expensive in terms of time and memory.

K-NN can be used for both classification and regression. In case of classification, we take the majority vote of k-nearest neighbours to find the label of a test data point. This is a lazy machine learning model in the sense that it defers all the work to the actual classification phase. During the training, only the feature vectors of the training emails are stored with their corresponding labels. All the vectors are imagined as points in a space with the distance being Euclidean distance.

Based on the assumption that similar things exist in proximity, all data points are assumed to be points in a space where similar data points with the least distance between them appear close to each other. Examples are shown below with red stars and green triangles in Figure 4.15. kNN implementation follows the steps: loading data, initialising k, transforming data points, calculating the similarity measure or distance of a novel input to the training data points, finding the close by neighbours and finally voting for the label. The labels are combined using a simple majority vote. For classification, this label is the mode (the label with high a number of occurrences) of k labels and for regression it is the mean of the k labels.

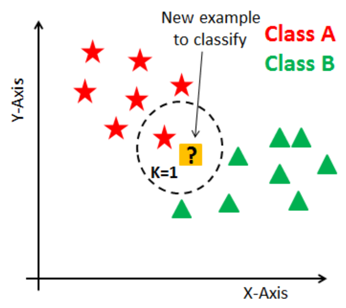


Figure 4.15: k-Nearest Neighbors algorithm with k=1 (Navlani, 2018)

The nearest neighbour k is a hyperparameter that must be defined at the time of model development. For even numbers of labels, also known as class values, an odd value of k is chosen based on the majority vote of its neighbours for the case of classification. For example, in Figure 4.15, for even label = 2, k holds an odd value = 1 in this example. To select the correct value of k, one method is to run the algorithm multiple times with different values of k and chose the one with the best performance. Research has demonstrated that there is no optimal number of neighbours - it depends on the dataset distribution and size. If the number of neighbours is too low, this influences results by introducing noise. For example, for k=1, the prediction is always accurate as the closest point to the training datapoint is the test point itself which is overfitting the boundary as in Figure 4.15. On the contrary, large numbers of neighbours as in Figure 4.16 make the computation expensive. However, the large number of votes makes the prediction more stable and it is more likely to be correct, although this starts to show errors as the value of k becomes larger, indicating that k has been stretched.

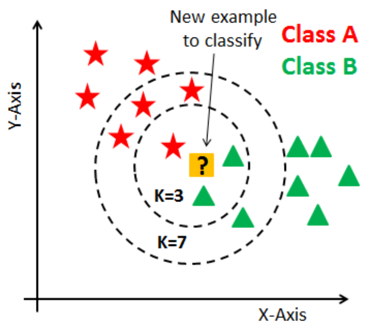


Figure 4.16: k-Nearest Neighbour algorithm with multiple higher values of k (Navlani, 2018)

The next step in kNN involves transforming data points into feature vectors followed by the calculation of distance between the mathematical value of these data points via a distance measure, a common one being the Euclidean distance function which is defined as

$$d(x,y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

where $(x_1, y_1), (x_2, y_2), \dots. (x_k, y_k)$ are the sample space and y is the class label for x.

The visual representation of Euclidean distance is shown in Figure 4.17. Other distance measures, such as Hamming, Manhattan and Minkowski distance, can also be used. It is useful to assign weights to neighbour contributions to ensure closer neighbours contribute more than those further away. The kNN further calculates the probability of these data points based on their similarity to the novel data and uses the highest probabilities for prediction.

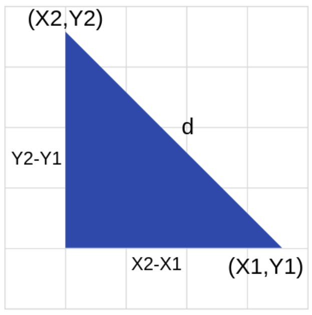


Figure 4.17: Euclidean distance representation

In the case of classification, the majority vote of k-nearest neighbors is considered to find the label for a test data point. For regression, the average of the values for k-nearest neighbours of a point are considered to find the label for a new test point. To make regression more accurate, the response of the nearest neighbour is given more weight than those further away. Thus, point weight decreases inversely with their distance from the test point.

When a new email is to be classified as ham or spam, first the K-nearest neighbors of this new email in the feature space is identified and then a majority vote is taken, i.e., the most frequent label (ham or spam) among the neighbours is used to label the new data point.

In case of text classification, sometimes a better choice of the distance metric would be Hamming distance rather than Euclidean distance. The hamming distance between two strings is the number of positions at which the two strings differ from each other.

Nearest Neighbors is a common clustering technique. Based on a set of data points, groups/clusters of points that go together are identified, i.e., appear in close proximity of each other as if they are forming a community. So, in this particular case, the two clusters that are expected to form from the points in the feature space of the emails are the spam and ham categories.

The K-Nearest Neighbors problem is NP-Hard and, hence, requires excessive amounts of computational resources for the absolute solution. In consequence, an approximation algorithm is used to find a solution for which Llyod's algorithm is commonly used for solving K-Nearest Neighbors. For this algorithm, the starting point is an initial assignment of points. Each point in assigned to a cluster, either spam or ham after which the centroid of the clusters is computed. With this new centre, the points are re-assigned to the clusters. This process is carried out iteratively to improve the cluster assignments. With convergence achieved when the newly computed centroids for the ham and spam clusters are the same as the previous. Hence, a limit is assigned on the number of iterations although that may mean the algorithm may never converge. kNN is one of the simpler techniques for prediction and is used for email spam filtering due to its low calculation time and ease of interpretation. Firte, Lemnaru, and Potolea (2010) and Chakrabarty and Roy, (2014) promote the use of kNN for email spam detection.

Logistic regression (LR) is one of the most commonly used supervised machine learning classification algorithms for discrete binary classification situations that works by maximizing likelihood, i.e., maximizing P[y|X] where X is the feature matrix (document term matrix) where each row is a feature vector and y is the vector of labels, one element for each row in X. Since the probability function is continuous and real as opposed to discrete, it is labelled logistic regression. The model is trained to learn probability distribution of the ham and spam over the set of attributes in the feature set.

LR is somewhat misnamed as it is used for classification. However, this statistical model builds a real continuous function with real valued attributes to predict the probability of data membership, in this case of a class. This function is also called the sigmoid function and is the probability of getting a label. Sigmoid is an S shaped curve that works by taking a real value as input and mapping it to any discrete value between 0 and 1 which then can be transformed to 0 or 1 depending upon the chosen threshold. LR uses a linear equation to represent the input values linearly combined with the weights to produce a binary output as prediction (Aggarwal, 2012). This is called logistic regression because the probability function is continuous and real as opposed to discrete as in the case of linear regression. The concepts learned in LR can be useful for deep learning using neural networks.

Training the model for logistic regression involves defining an error measure, in this case it is 'likelihood', i.e., how likely is it that to generate the training responses as spam and ham from the training features. Once the error measure is defined, learning the model is translated into an optimization problem wherein the error measure is to be reduced while changing the variables that it depends upon.

Suppose X is the input real value represented as linear equation in the form of $X = ax + b$, with coefficients b as bias and a as the constant, then the sigmoid (logistic) function to map the prediction to probability is represented as

$$S(x) = \frac{1}{1 + e^{-x}}$$

where $S(x)$ is the output between he values of 0 and 1 shown in Figure 4.18 and *e* is the natural log base. For positive values of x, the logistic function becomes asymptote to output as 1 and to 0 for negative values.

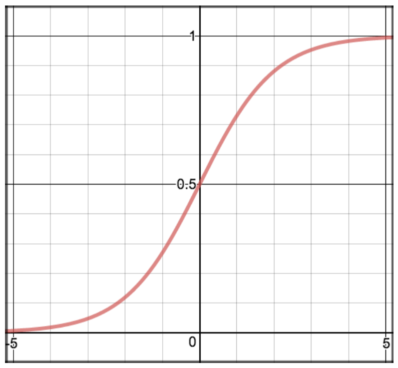


Figure 4.18: A Typical logistic(sigmoid) function

LR predicts probabilities by modelling the probability of the default label; the coefficients for the input values are determined from the training data. This is achieved through maximization of the likelihood that a novel data point is classified correctly, i.e., maximizing $P[y|X]$ where X is the feature matrix, each row of which is a feature vector and y is the vector of labels, with one element for each row in X. This is called maximum likelihood estimation which is an approach for parametric estimation in statistical methods. Methods such as Newtons method or Gradient boosting can be used for maximising likelihood. The model is trained for learning the probability distribution of labels over the set of attributes. Training a model for logistic regression involves defining an error measure. An error is a value where magnitude identifies how far it deviates from the learned model that would predict correctly from the training data. For email spam filtering problems, given the output classes are 2, logistic regression is binomial, with y holding two labels, 1 for email as spam and 0 for legitimate emails. Given that LR is capable of binary classification, it is used in this thesis. Thus, the two probabilities can be defined as

$$P(y = 1|X) = S(v)$$
$$P(y = 0|X) = 1 - S(v)$$

The chosen threshold value acts as a decision boundary above which the value belonging to one label and below to the other. For example, if the threshold value were .5, and the prediction function returned a value of more than .5, the novel input email would be classified as spam.

The error measure in LR is defined as a cost function which is created and minimised in order to develop an accurate model with minimal error. The cost function is defined as

$$-log(h(x))\ if\ y = 1$$
$$-log(1 - h(x))\ if\ y = 0$$

where h is the hypothesis that probabilities lie between 0 and 1. The two can be compressed into one overall cost function J(θ) as

$$J(\theta) = -\frac{1}{2}\sum[y\ log(h_\theta(x) + (1-y)\ log(1-h_\theta(x))]$$

To minimise the cost value, gradient descent is used on each parameter to minimise $J(\theta)$.

Logistic Regression has been used individually and as part of hybrid and ensemble models for email spam detection with accuracy between 91-98% (W.Wang, 2010; Wijaya & Bisri, 2016; Yang, Liu, Zhou & Luo, 2019).

A Bagging Ensemble of CART, LR kNN, and SVM (CLKS):

Bagging ensemble use mean, weighted average or majority vote to combine the output of the independent predictions made by different classification algorithms as shown in Figure 4.19.

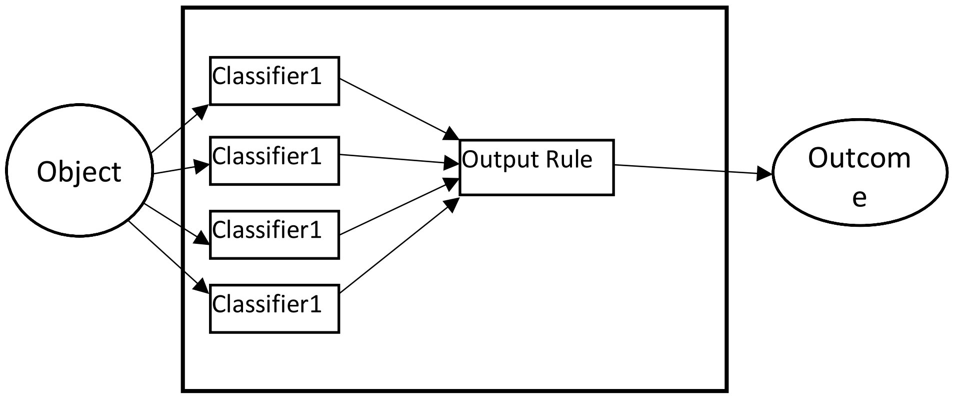


Figure 4.19: Bagging Classifier Ensemble

This bagging ensemble combines output of CART, kNN, SVM and LR, and produces single prediction output using a voting mechanism.

M. Iqbal, Shoukat, Khan, and Iqbal (2011) which presents a performance analysis of k-Nearest Neighbour and Naïve Bayes algorithm classifiers for email spam filtering. Design aspects of both classifiers have been presented in terms of computational complexity and accuracy of classification. SVM are an example of the kernel method which can be applied as linear or gaussian kernel method, which are important areas of machine learning theory. Its accuracy can be further improved by using it with bagging and boosting as an ensemble. Suryawanshi, Goswami, and Patil (2019) present an ensemble classifier based on Naïve Bayes, Support Vector machine, k-Nearest neighbour and Bagging & Boosting machine learning algorithms. The results show high performance in low false positive rate and high accuracy using dataset from UCI machine learning repository.

Harisinghaney, Dixit, Gupta, and Arora (2014) used Naïve Bayes, Support Vector Machine and density-based spatial clustering of applications (DBSCAN) algorithm to with an aim to detect email spam based on text as well as image. Pre-processing of the email datasets prior to classification is performed to increase the prediction capability. S. Ali (2019) developed a model with SVM and k-NN to maximise the benefits from the differing characteristics of the two. Where SVM prepares the features to learn from the examples from the information set during training and in case an example is not selected, kNN has a different strategy for example determination based on similarity and proximity estimation. Hence, kNN discovers close neighbours to question set and SVM jellies the separation on gathering those neighbours, results showed 98% correct prediction on image spam.

Gashti (2017) carried out a comparative analysis for email spam detection using a combination of Harmony Search algorithm and Decision Trees for feature engineering on Spambase dataset using models Bayesian Additive Regression Trees (BART)(Abu-Nimeh, Nappa, Xinlei, & Nair, 2008), Random Forest, Classification and Regression Trees (CART), SVM, NB and Logistic Regression (LR). Datasets Ling-spam and PU1 were used for assessment, evaluation and comparison, the results showed that accuracy for Random Forest at 98.61% was highest on Spam Base dataset whereas accuracy was 99.8% on Ling-spam and 97.12% PU1 datasets, the maximum error rate reduction was shown by CART decision tree algorithm at 2.2% in comparison with BART. It is noticed that with appropriate feature engineering, LR performs accurately as shown in this case.

The rules are defined by the following voting system to classify a new email message:
[all four methods agree] : outcome is the agreed classification decision
[Three methods agree]: outcome is the agreed classification decision
[Two methods agree]: outcome is Ham. This outcome has been chosen with an aim to reduce FP as some degree of FN is acceptable whereas FP is not acceptable

4.8 Experimental Results – Dynamic Multi-Layer Ensemble Model

To evaluate the performance of the dynamic multi-layer ensemble model, a series of experiments were conducted on the datasets identified in Table 4.2 and compared with the outcomes of the dynamic multi-layer model. The experiments were conducted to test the multi-layer model at two levels. At the first level, experiments were conducted to test the performance of the multi-later ensemble model using the datasets. At the second level, after integrating the multi-layer model with the Bayesian classifier, the greys are classified and the 'spams' reclassified and moved to the appropriate folder by the Bayesian classifier. Once a satisfactory level of performance was achieved with the multi-layer model on its own, the integrated DMLEM Framework (Figure 4.12) was evaluated.

The feature selection was carried out for semantic features using BoW with term frequency and for syntactic features using the same methods described in Section 5.4 for all 13 features. A Chi-square test was employed to select the most significant features. In addition, a document term matrix containing feature vectors was developed. The models were generated iteratively to identify the best parameters for performance of each of the models using the training samples. Once trained, the evaluation of each of the models in CLKS ensemble was performed using the test samples for all datasets. Each dataset contained samples of emails belonging to the two output classes of spam and ham. The datasets were divided into training and testing samples at a 70:30 ration respectively. The 70 % training samples were selected randomly and the remaining 30% were left for testing. The datasets were chosen to belong to different times and sizes of the datasets differed. The final prediction for the email message as spam or ham was then recorded as the outcome of DMLEM.

Experiments were conducted to extract the selected optimum number of 1000 semantic features from the training samples for each of the datasets. From the same training samples syntactic features for all 13 elements determined in Section 4.4 were selected. Then, the model was generated and trained using a total of 1026 features which were developed based on selected features from header and body of the email samples.

In the testing stage, CLKS and DMLEM were evaluated on the test samples containing both spam and ham emails. Table 4.11 shows the features initially extracted as bigrams from the training samples, the time taken for extracting those features, the speed for initial feature selection, time taken to extract the most significant features using the chi-square test, training and testing times for the DMLM model. From these statistics it is evident that by choosing bigrams it has not increased computational complexity or time. The time taken to train and test the model on semantic and syntactic features has also not impacted the performance time of the classification model. The time durations for training and testing for the DMELM with ensemble of CLSK was efficient in terms of computation time.

Table 4.11: Comparison of DMLM and DMLEM for computation complexity

Dataset	Number of Sementic eatures (bigrams)	Feature extraction duratiom (seconds)	Significant Feature selection duratiom (seconds)	Speed (MB/sec)	DMLM Train time	DMLEM Train Time	DMLM Test Time	DMLEM Test time
CSDMC2010	728598	15	0.91	1.27	53	83.6	17	20
ENRON1	247949	3.84	0.28	1.13	58	95.2	24	26
ENRON2	318353	5.8	0.32	1.1	64	102.2	21	24
ENRON3	470345	9.1	0.43	1.1	69	107.2	25	30
ENRON4	329480	4.7	0.26	1	72	102.8	22	31
ENRON5	272177	4.5	0.27	1.1	72	98.57	28	36
ENRON6	368384	5.9	0.33	1.05	66	111.3	25	30
LINGSPAM	419467	6.2	0.31	0.9	65	56.1	19	23
PU1	152526	2.6	0.14	0.75	13	21.5	15	20
TREC07	599843	18.2	0.96	1.03	72	80.8	24	29

Evaluating the performance of the machine learning algorithm is an essential part of successful classification. Most existing experiments have used accuracy of classification as the indicator of performance. However, measuring accuracy in isolation may be misleading and evaluation of the classification model is more reliable when assessed through a range of performance measurements such as false positive and false negative rates, precision and recall. Earlier section used accuracy, FP rate, and FN rate for performance evaluation. These measures have a technical interpretation associated with the outcome values obtained (Flach, 2019). For classification, there are two types of outputs generated: class output and probability output. For binary classification, the class outputs can be either 1 or 0. Example algorithms that produce a class output are SVM and kNN. Probability outputs are given by algorithms such as logistic Regression, Naïve Bayes, these probabilities can be converted into classes by defining threshold probability values for the output classes.

The performance of the DMLEM was measured for accuracy, precision, recall, FP and FN. The results from the DMLEM are compared with those of the DMLM and the Bayesian classifier to determine which classifier showed superior performance. In terms of the FP and FN rates and accuracy between Bayesian classifier (at two threshold value sets), DMLM and DMLEM for all datasets is presented in tables and graphs in this section while the analysis of the results is presented in next section in 'Analysis of results and Discussion'

Experimental results are shown in Table 4.12 presenting FP and FN rates and accuracy measures for the DMLEM from the results obtained from classification for the test samples of the datasets. The table also offers a comparison between the results obtained from the experiments for Bayesian classifier and DMLM.

Table 4.12: Table showing False Positive rate, false negative rate and accuracy figures for Bayesian classifier 0.15-0.09 threshold, Bayesian classifier 0.5-.5 threshold, DMLM and DMLEM all 10 datasets

DATASET	METRIC	BAYESIAN CLASSIFIER 0.15-0.9 Threshold	BAYESIAN CLASSIFIER 0.5-0.5 Threshold	DMLM CLASSIFER	DMLEM CLASSIFER
CSDMC2010	FP RATE	0	0.23	0	0
	FN RATE	0	1.69	0.7	0.3
	ACCURACY	100	98.08	99.3	99.7
ENRON1	FP RATE	1.3	2.7	1	0.9
	FN RATE	0	3.9	1.4	0.2
	ACCURACY	98.7	93.4	97.6	98.9
ENRON2	FP RATE	0	1.1	0.2	0.2
	FN RATE	0.1	5.02	1.4	0.3
	ACCURACY	99.9	93.88	98.4	99.5
ENRON3	FP RATE	0.1	1.36	0	0.06
	FN RATE	0.1	5.7	1.3	0.6
	ACCURACY	99.8	92.94	98.7	99.34
ENRON4	FP RATE	0.1	4.7	0.1	0.1
	FN RATE	0.3	0	0.3	0.3
	ACCURACY	99.6	95.3	99.6	99.6
ENRON5	FP RATE	0.1	7.2	0.4	0.4
	FN RATE	0.2	0.89	0.2	0.1
	ACCURACY	99.7	91.91	99.4	99.5
ENRON6	FP RATE	0.2	8.8	0.7	0.6
	FN RATE	0.3	0.07	0.3	0.2
	ACCURACY	99.5	91.13	99	99.2
LINGSPAM	FP RATE	0	0.72	0.1	0.1
	FN RATE	0	9.3	1.3	0.1
	ACCURACY	100	89.98	98.6	99.8
PU1	FP RATE	0.9	3.09	1.2	1.5
	FN RATE	0	2.16	1.2	0.3
	ACCURACY	99.1	94.75	97.6	98.2
TREC07	FP RATE	0	1.1	0	0
	FN RATE	2.9	0	2.9	2.9
	ACCURACY	97.1	98.9	97.1	97.1

As can be seen from the table the percentage of correct classification for DMLEM has improved by including the CLSK ensemble for classification.

A comparative graphical representation of the FP and FN rates are shown in Figure 4.20. For the FP rate, the performance of DMLEM was superior to all other classifiers while for the FP rate, the DMLEM accounted for some improvement for some datasets, but had no impact on some others and deteriorated in terms of the PU1 dataset.

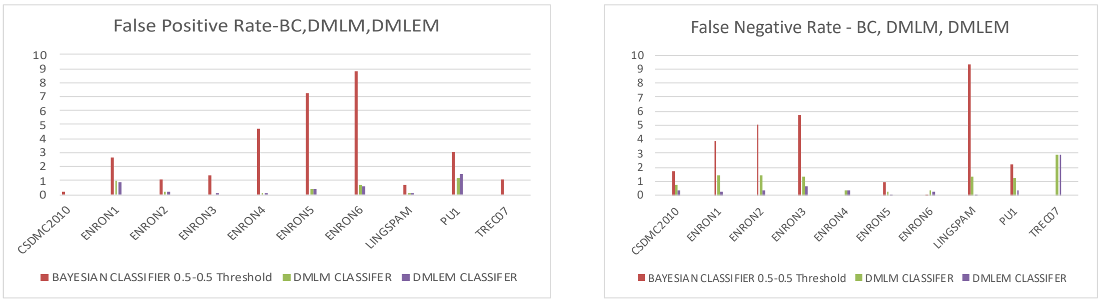


Figure 4.20: Graph showing a) false positive and b) false negative rate for all 10 datasets for Bayesian classifier 0.5-.5 threshold, DMLM and DMLEM

The classifiers were measured for the detection rate - Figure 4.21 shows a comparative representation of the three models. It is evident that the accuracy of the DMLEM is always higher than that of the Bayesian classifier and higher than the DMLM, except for the ENRON datasets.

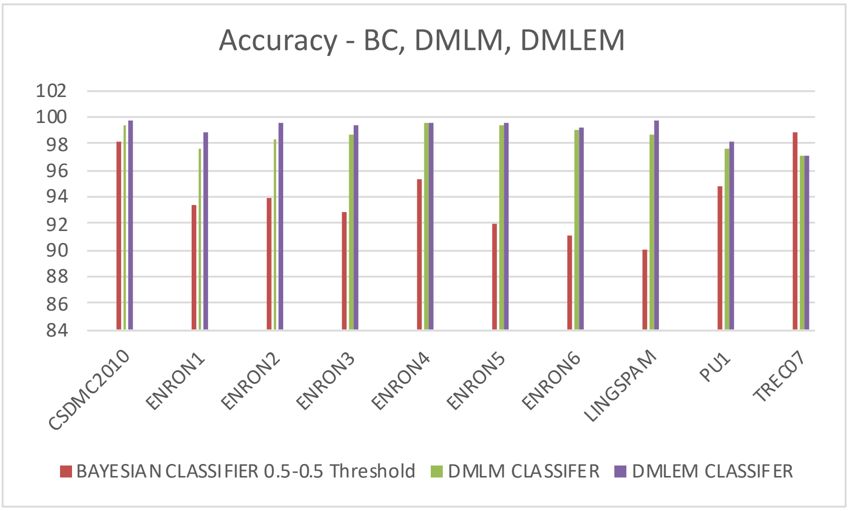


Figure 4.21: Graph showing accuracy comparison for Bayesian classifier 0.5-.5 threshold, DMLM and DMLEM for all 10 datasets

4.9 Analysis of Experimental Results and Discussion

The results in Table 4.12 reveal that semantic and syntactic features provide sufficient knowledge to the CLSK ensemble and the proposed DMLEM model to achieve satisfactory overall detection performance. In this section, the information presented in the tables is further analysed.

Table 4.13 shows the comparison of accuracy for Bayesian classifier at the two threshold levels with DMLM and DMLEM.

Table 4.13: Comparison of Accuracy

DATASET	BAYESIAN CLASSIFIER 0.15-0.9	BAYESIAN CLASSIFIER 0.5-0.5	DMLM CLASSIFIE R	DMLEM CLASSIFIE R
	Accuracy	Accuracy	Accuracy	Accuracy
CSDMC2010	100	98.08	99.3	99.7
ENRON1	98.7	93.4	97.6	98.9
ENRON2	99.9	93.88	98.4	99.5
ENRON3	99.8	92.94	98.7	99.34
ENRON4	99.6	95.3	99.6	99.6
ENRON5	99.7	91.91	99.4	99.5
ENRON6	99.5	91.13	99	99.2
LINGSPAM	100	89.98	98.6	99.8
PU1	99.1	94.75	97.6	98.2
TREC07	97.1	98.9	97.1	97.1

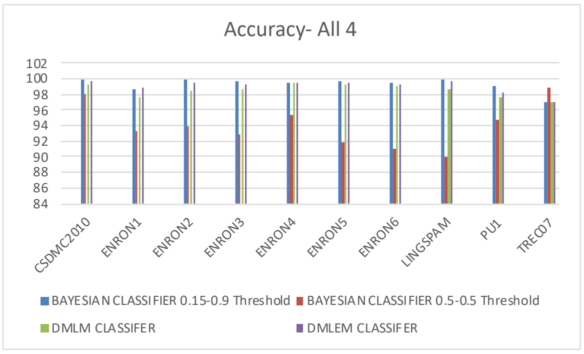


Figure 4.22: Accuracy comparison among different models

The results show that DMLEM outperforms the other two classifiers at the same threshold levels except for the TREC07 dataset where the accuracy is 98.9% whereas both DMLM and DMLEM have 97.1%. It is important to note that TREC07 dataset size is much larger; however, only a smaller portion of this dataset was chosen for these experiments so that the difference in dataset size does not create a bias in the experiments. Nevertheless, reducing the dataset size through random selection of samples, may well have caused the loss of some samples with syntactic attributes impacted the feature selection for the DMLM and DMLEM models.

Compared with the Bayesian classifier at 0.5-0.5 threshold, both the DMLM and DMLEM have significantly improved accuracy for all datasets with a mean difference of 5% increase in accuracy except for CSDMC2010 and TREC07 dataset. This is interesting since both of CSDMC2010 and TREC07 datasets belong to similar time periods.

The size of the original Enron dataset is large; thus, as for TREC07, subsets were extracted from the Enron dataset to keep all dataset to a comparable size. The performance of the DMLM model on the Enron dataset significantly improved compared to the Bayesian classifier. The feature selection in the Bayesian classifier is achieved using a basic BoW whereas for the DMLM, an improved BoW selected semantic and syntactic features. The detection rate for the DMLEM improved compared to the DMLM for all datasets except for TREC07 dataset which means that use of a bagging ensemble has proven efficient.

The performance results for the Bayesian classifier with far apart thresholds for its best performance have been included in this analysis to determine if the performance of the proposed model is better or worse in comparison. Figure 4.23 shows that the performance of the DMLM and DMLEM is comparable to that of Bayesian classifier with 0.15-.09 thresholds; it is important to note, however, that at those threshold values, the Bayesian classifier leaves significant amounts of unclassified emails which include misclassifications. Hence, overall, the DMLEM proved to be have the highest detection rate.

Compared to the DMLM, the CLSK ensemble does improve the FP rate of the DMLEM slightly for some datasets as shown in Table 4.14 but remains constant with the DMLM for many others and for PU1, it is slightly higher for the DMLEM compared to the DMLM but much lower than compared to Bayesian classifier at the same threshold value. Figure 4.23 shows the efficiency in FP rates for the DMLEM.

Table 4.14: Comparison of rate of False Positive

DATASET	BAYESIAN CLASSIFIER 0.15-0.9 Threshold	BAYESIAN CLASSIFIER 0.5-0.5 Threshold	DMLM CLASSIFER	DMLEM CLASSIFER
	FP RATE	FP RATE	FP RATE	FP RATE
CSDMC2010	0	0.23	0	0
ENRON1	1.3	2.7	1	0.9
ENRON2	0	1.1	0.2	0.2
ENRON3	0.1	1.36	0	0.06
ENRON4	0.1	4.7	0.1	0.1
ENRON5	0.1	7.2	0.4	0.4
ENRON6	0.2	8.8	0.7	0.6
LINGSPAM	0	0.72	0.1	0.1
PU1	0.9	3.09	1.2	1.5
TREC07	0	1.1	0	0

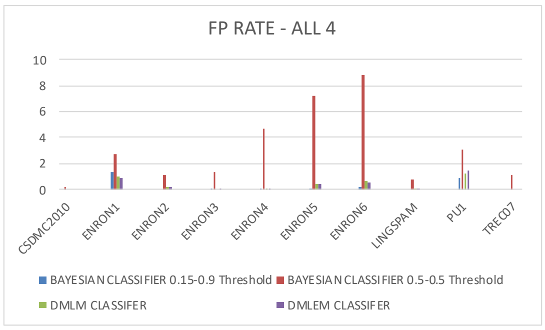


Figure 4.23: False Positive rate comparison

One of the concerns in terms of the DMLM, were slightly higher FN rates which was one of the motivating factors for introducing the CLSK ensemble in the DMLM. The results in Table 4.15 show improved rates for FN rates for all datasets except for TREC07.

Table 4.15: Comparison of rate of False Negative

DATASET	BAYESIAN CLASSIFIER 0.15-0.9 Threshold	BAYESIAN CLASSIFIER 0.5-0.5 Threshold	DMLM CLASSIFER	DMLEM CLASSIFER
	FN RATE	FN RATE	FN RATE	FN RATE
CSDMC2010	0	1.69	0.7	0.3
ENRON1	0	3.9	1.4	0.2
ENRON2	0.1	5.02	1.4	0.3
ENRON3	0.1	5.7	1.3	0.6
ENRON4	0.3	0	0.3	0.3
ENRON5	0.2	0.89	0.2	0.1
ENRON6	0.3	0.07	0.3	0.2
LINGSPAM	0	9.3	1.3	0.1
PU1	0	2.16	1.2	0.3
TREC07	2.9	0	2.9	2.9

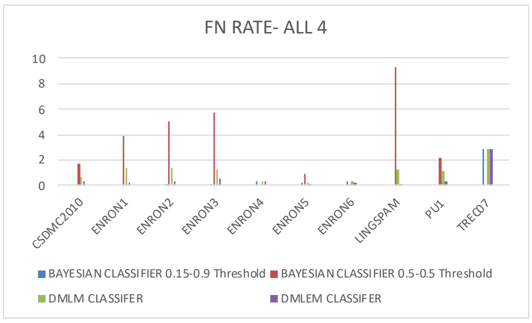


Figure 4.24: False Negative rate comparison

Compared with the Bayesian Classifier, the multi-layer models achieved significantly higher detection rates as well as lower FP and FN rates. Hence, it can be concluded that the improved feature selection methods were efficient in improving the classification performance.

The % improvement for each of the datasets was calculated for overall spam detection which includes ham and greys for the Bayesian classifier that contributes towards FP. DMLEM has shown 1.65-11% improvement from over the Bayesian classifier model and up to 1.3% improvement in performance for the spam detection rate.

To further compare these results of multi-layer ensemble model, a comparison to other models was carried out and presented in Table 4.16. Based on this evaluation, it is clear that the DMLEM is efficient in performance when compared to other ensemble classifiers. The number of features that were included in the DMLEM cover a great variety inclusive of spammer technique, at the same time not compromising on computational complexity and cost as shown on Table 4.11.

Table 4.16: Comparison of detection rate for DMLEM with other Ensemble models

Ensemble Reference	**Machine Learning methods Used**	**Dataset Used**	**Accuracy**	**Features**
1 (Anandita et al., 2017)	Bernoulli Naïve Bayes, Random Forest, k Nearest neighbour, Support vector machine	Spam Assassin	96%	Text from body and header
2 (Varghese & Dhanya, 2017)	AdaBoost, Random Forest, Support vector machine	Enron	93.60%	text from body only
3 (A. Wijaya, & Bisri, A., 2016)	Logistic Regression, Decision Tree	Spam Base	91.67	Text features only
4 (Chharia & Gupta, 2013)	Naïve Bayes, CART, C4.5, ADT, Random Forest,	Enron Spam Assassin	96.4. 98.6	Text (5 features) + Non-text-(5 features)
5 (S. K. Trivedi & Dey, 2013)	Genetic Programming, Adaptive Boosting	Enron Spam Assassin	94.1. 98.6	Text from body and header
DMLEM	CART, Support vector machine, Logistic Regression, k Nearest neighbour	Enron, CSDMS2010, TRCE07, PU1, Ling-spam	99, 99.3, 97.1, 98.2, 99.8	Text (1000 features) + Non-text-(26 features) from header and body

The results showed that the multi-layer ensemble model improves the performance of email spam classification and detection by reducing the overall % of FP and FN. This means that THE model is performing at 99.8% which is an encouraging improvement.

Chapter 5

User Profiling for User Preferences to Personalise Email Classification

I use Spam Arrest because of the amount of junk mail I get. Any legitimate person who wants to send me a message has to jump through hoops before they can be added to my opt-in list.
-Kevin Mitnick

5.1 Overview

From the previous Chapters 3 and 4, it is evident that individual user preferences differ which is an important factor to consider while filtering email spam. The key issue in email spam filtering is the high number of false alarms, especially FP due to the fact that the employed classification models consider unfamiliar normal behaviour as spam. If a user does not normally get emails of a kind and now such emails are being received, then these may perhaps be spam (unwanted) for them. The quality of training data plays a vital role in teaching filters what spam and ham are for a user. In Chapter 4, experiments were conducted with context and user specific data for training of the Bayesian filter with results showing that such training of the classification models does significantly improve the classification detection rate. In Chapter 4, a framework was developed to add additional layers of filtering at the client side to provide high classification accuracy for email spam for the user.

One step further from the context are a user's individual interests and preferences. Clearly filters need instructions to identify which email is spam and which is ham, in particular for a user. In some mail clients, such as Gmail, users can exercise their preference through a setting that allows them to choose the area of interest which the filter uses to classify incoming emails. This is time-consuming for the user and requires continuous updating of preferences as users' interests change frequently. Hence, there is a need for a filter to learn and use this learning to change the classification of emails according to the interest of the user. Little research has been devoted to considering specific user preferences while designing anti-spam filters. H.-J. Kim, Shrestha, Kim, and Jo (2006) suggest user action based adaptive learning where they attach weights to Bayesian classification on the basis of user actions. Jongwan Kim, Dou, Liu, and Kwak (2007) constructed a user preference ontology on the basis of user profile and user actions and trained the filter on the basis of that ontology. All of these suggested models are based on user actions related to emails and email spam for classification. However, none of the works has so far addressed user context and preferences.

The filter takes user data as input to be fed into the email spam filter model for training and model development. Such a model would be productive for the users, improve the performance of the filter and cater for user specific needs.

User profiling aims to understand the preferences of a user by applying machine learning techniques to learn from the users' data. This is achieved through feature extraction from the data of an individual user and building models and training them using appropriate samples. Then, when a new email arrives, the features extracted for that email are matched with the features identified for that user to classify it as ham or spam. So, if a user likes romance and animation movies and receives emails related to war movies then it is potentially spam (unwanted) for that user.

Another benefit of user profiling is that most content-based filters at organisational level are generic and are applied to all the users. These filters remove some emails that are mass sent; however, they may be ham (wanted) for a particular user but may be allocated to 'junk' because the mail server filter identified it as spam. To avoid such occurrences, user profiling may include a step where the spam folder at the client side is re-classified to bring those FP back into inbox.

An important consideration is that the filter training should not require minimal manual input from the user.

5.2 User Profiling for User Preferences - User Behaviours and Actions?

Content-based email spam filters are employed at the mail server of organizations and the rules of filtering are applied to emails for all different users irrespective of their interest. However, as discussed earlier, individual users have different preferences in terms of what they want to receive in their mailboxes. The same email that may be classified as spam by one user, may be ham for another and vice versa. An example comes from academics at a university who may be interested in different topic areas. A book publisher emailing academics about newly published titles may be seen as useful by an academic who works in that area whereas it is spam to another who does not. Hence, to address individual needs of users, there is need to update the email spam filter to incorporate the user's preferences.

Over the last few years, this area of research has come to be of interest as personalized spam filtering where filters adapt to individual user preferences (Cheng & Li, 2006; K. N. Junejo, & Karim, A., 2013; Li & Shen, 2011; X. Liu et al., 2017; Zhijun, Weili, Na, & Lee, 2005). Personalized filters can be deployed at the server or client side. Yue, Abraham, Chi, Hao, & Mo

(2007) presented a user feedback-based artificial immune system at the server level, whereby servers dynamically adapt to behaviour-based patterns of individual users within the same email server. This method uses a clustering-based approach in conjunction with other filtering systems for error minimization in email spam classification. This study bridged the gap between global filtering systems and individual user preferences. The main issue with this technique is that it requires users to provide feedback and secondly, the model is dependent on the accuracy and correctness of this feedback.

Literature suggests that most of the proposed solutions are at the server side (Cheng & Li, 2006; K. N. Junejo, & Karim, A., 2013; Li & Shen, 2011; X. Liu et al., 2017; Zhijun et al., 2005). However, these solutions cannot be effectively implemented for the following reasons. Firstly, these solutions have to be robust and lightweight since it is to be deployed for thousands or millions of users by email service providers, else it is not practical to implement it. Secondly, it requires a significant amount of organizational, human and technical resources to implement and maintain such solutions. Thirdly, the computational cost associated with personalized filters is extremely high at the server side and may make the filtering process slower.

A solution would, therefore, be to shift the fine-grained filtering process to the user where cost is confined to adjustments needed for one user. Kim et al. (2007) have argued for considering the user's behaviour as the basis for any action taken on email spam such as determining the destination of an email. However, so far, limited effort has gone into user behaviour-based email spam filtering although some studies exist. One of these comes from Alsowail and Batarfi (2011) who studied the use of an adaptive email filtering learning system based on user behaviour and a rating system that allocates rates to user actions carried out on an email. A similar weighted system was proposed by Han et al. (2008) for detecting email spam based on user actions, where weights were allocated to each user action. Viewing Grey lists was identified by Isacenkova and Balzarotti (2014) as user behaviour in the sense that the users viewing grey emails was considered as whitelisting them. The statistics presented indicated that on average 3.9 emails from a legitimate email campaign were contained in the grey lists compared to 1.1 emails from a spam campaign. The limitations in this case concern the bias introduced due to the possibility that a user accidentally views a spam emails which would impact classification.

This research argues that user actions cannot be taken as a determining factor to decide whether an email is spam or ham. To argue this point, user actions associated with ham and spam emails are presented in Table 5.1 and 5.2 respectively.

Table 5.1: User Behaviour Associated with Ham Emails

1. Open, Read and Delete	some emails are opened, read and delete as they are not needed for later use.
2. Open, Read and Save	some emails are read and saved for later use.
3. Open, Read, Reply and Delete	some emails are read, replied and deleted as not needed for later reference.
4. Open, Read, Reply and Save	some emails are read, replied and save for later reference.
5. Delete	user deletes the email without opening, it is not relevant to the user.

Table 5.2: User Behaviour Associated with Spam Emails

1. No Action	user ignores the email, and it stays in the user inbox as spam
2. Open and Delete	user opens and then deletes the email.
3. Open, Read and move to Spam folder	user opens, evaluates and then moves the email to Spam folder.
4. Move to spam folder	user moves the email to spam folder without opening it
5. Blocked.	User blocks the email to spam folder without opening it
6. Open, Read and Blocks	user opens, evaluates and then blocks the email.
7. Delete	user deletes the evident spam email

A deeper analysis of user actions from the Tables 5.1 and 5.2 reveals that a user may carry out a mix of similar actions on several emails sent by the same sender. The emerging picture, however, is not comprehensive as these actions are invariably carried out on ham and spam emails. Hence, user behaviour via user actions associated with an email is not a clear indicator of users' preferences in terms of email spam as suggested by most of the research papers.

Pham et al. (2011) developed an architecture for proxy servers for collaborative spam filtering that improves the performance of SMTP servers. This is based on feedback received from the users' email clients from collecting user feedback when an email is incorrectly classified by the filter at the server level and monitoring the actions of the users. The agent extracts the terms from the emails *deleted* by the user without reading and the user profile is updated based on such user action. From Table 5.1, it is clear that a user might *Delete* an email from a sender without opening in the case when it is not relevant to them at that time and hence this action is not a correct indicator of an email as spam. If based on action, the content-based filter is updated with the words found in that email as indicators of spam which would mislead the content filter and may lead to increasing the FP loss of information that is important for the user.

Analysing the user behaviour from the data logs of the user emails and social networking terms of user interactions would allow a reasonable insight of user preferences (Jiang, Jin, Yuanyuan, and Guandong, 2016). Such analysis of the interactions of a user with several other users and sender/receivers of emails could provide useful information about the users' preferences. Knowledge of these preferences can clearly increase the performance of email spam filters.

None of these current solutions, however, are comprehensive, leaving a gap between what is offered and what could be achieved by other means. There is a need, therefore, to modify the email filter to incorporate user profiling – for instance, if the mailbox is the university mail, then any email about sex change, selling drugs, advertising porn sites, etc. is spam (with 100% certainty). Also, any email about invitation to be a referee, to sit on PC of a conference, etc. is not spam (with 100% certainty). Hence, with user profiling, the FN (spam) that escape the mail servers would be filtered at the client end according to user preferences.

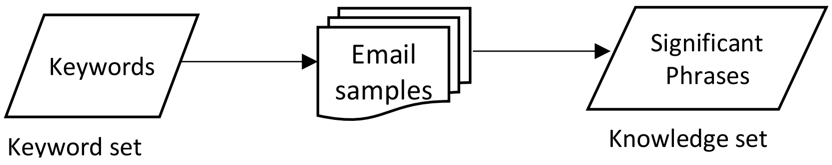
As discussed earlier, filters used to filter spam emails can be located at the sender server, receiver server or email client of the receiver. From the analysis above, there is justification to assume that filtering email with user preferences at the email client would yield better results, which justifies the research in later chapters of this thesis.

5.3 Methodology:

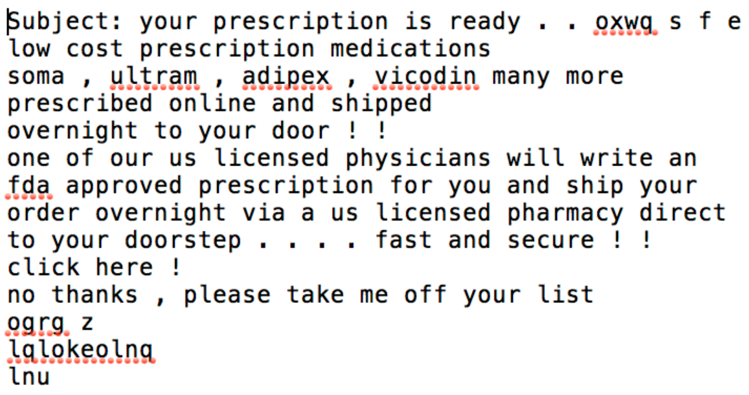
An iterative learning algorithm has been adapted for developing user profiling, purely developed based on the user data requiring no input from the user. Hence, no user logs or actions were included in the experiment. As a first step, a knowledge set is developed by identifying significant features from the user data in the form of significant phrases that provide meaningful information. It has been first assumed and then proven in Chapter 4 and 5 that multi words provide more accurate evidence rather than a single keyword. The frequency of a single keyword may be high, but when combined with another word they convey meaningful information pointing to a concept. Hence, it was decided that for user profiling significant phrases would be used, formed by locating the phrases around the keywords from the predefined keyword sets. A collection of significant phrases was developed through semantic analysis of the training samples of ham and spam emails.

It is assumed that each of these significant phrases express an opinion about the output class/label of an email as ham or spam. It is further assumed that significant phrases for different users differ based upon their preferences. This can be quantified by discrimination information measures.

For example, sample from two users spam emails is shown as i^{th} email for user 1and j^{th} email for user 2 in figure 5.1 a) and 5.1 b) respectively., after pre-processing and cleaning of data following phrases would be selected, for ith email - ‘prescription ready’, ‘low cost’, prescription medication’, etc. and for jth email – ‘major stock play’, ‘contract announcement’, ‘huge newsletter coverage’, etc.

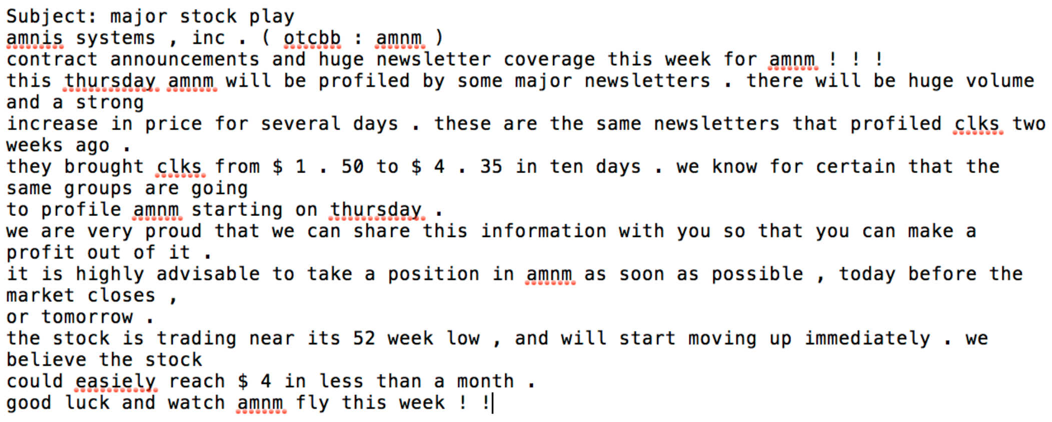


a) Significant phrases from emails



```
Subject: your prescription is ready . . oxwq s f e
low cost prescription medications
soma , ultram , adipex , vicodin many more
prescribed online and shipped
overnight to your door ! !
one of our us licensed physicians will write an
fda approved prescription for you and ship your
order overnight via a us licensed pharmacy direct
to your doorstep . . . . fast and secure ! !
click here !
no thanks , please take me off your list
ogrg z
lqlokeolnq
lnu
```

b) ith email from user 1



```
Subject: major stock play
amnis systems , inc . ( otcbb : amnm )
contract announcements and huge newsletter coverage this week for amnm ! ! !
this thursday amnm will be profiled by some major newsletters . there will be huge volume
and a strong
increase in price for several days . these are the same newsletters that profiled clks two
weeks ago .
they brought clks from $ 1 . 50 to $ 4 . 35 in ten days . we know for certain that the
same groups are going
to profile amnm starting on thursday .
we are very proud that we can share this information with you so that you can make a
profit out of it .
it is highly advisable to take a position in amnm as soon as possible , today before the
market closes ,
or tomorrow .
the stock is trading near its 52 week low , and will start moving up immediately . we
believe the stock
could easiely reach $ 4 in less than a month .
good luck and watch amnm fly this week ! !
```

c)

Figure 5.1: Significant phrases from emails and sample emails

If the ith email is defined by the feature vector $v_i = \{fv_{i1}, fv_{i2}, \dots fv_{in})$ with $fv_{ij} >= 0$ is the jth attribute value and n is the size of the feature vector space. A significant phrase i in the knowledge-set is likely to be a spam term in a feature vector, if its probability in email spam training samples $p(fv_i/y = +)$ is more than its corresponding probability value in ham email training samples $p(fv_i/y = -)$. It is a significant feature if

$$p(fv_i/y = +)/\,p(fv_i/y = -) > pf \text{ for spam and} \quad (5.1)$$

$$p(fv_i/y = -)/\,p(fv_i/y = +) > pf \text{ for ham,} \quad (5.2)$$

where pf is the parameter for feature selection.

Feature selection methods were employed to extract the significant features based on the identified significant phrases from the training samples which was then converted to feature vectors. A feature vector space of size n was selected from each sample set.

The process followed in terms of methodology to profile users is as below.

1. Loading Data (Retrieving File for spam/ham mails): This step retrieved the spam and ham emails from the dataset, loaded the file for each mail and stored them as an array.
2. Pre-processing and representing Data (Data Cleaning): Here the cleaning of the data was performed in preparation for training. It involved removing punctuation marks, stop words that do not contribute much to classification, common words and white spaces etc. Contents were then reconstructed after all the above steps satisfactorily clean the data in preparation for training.
3. Feature Extraction (Tokenizing Data): This step involves breaking up the text of an email into words, performing stemming and lemmatisation to bring the derived or infected words to their stem or root forms. Based on the keyword sets, all possible combinations of phrases are found from the emails based on the keywords present in the training samples. The value of phrases is determined by using word embedding techniques such as term frequency, term frequency inverse document frequency and word2vec embedding where all of them convert phrases to feature vectors of numbers.

Term frequency is calculated by dividing the number of occurrences of a phrase or word in a document by the total phrases/words in that document, TF-IDF stands for "term frequency-inverse document frequency", is a term weighting scheme that reflects the significance of a token in a document contained in a corpus; word2vec is a shallow neural net that is used to produce word embeddings from text as input and feature vectors as output that present words in that input text. The features with the highest score are chosen as sample features for training and classification. Once phrases are ordered, the highly ranked ones are significant phrases selected as features.

A phrase is a spam phrase if its probability in the training samples of spam emails $p(fv_i/y = +)$ is higher than its probability in ham training samples $p(fv_i/y = -)$. A phrase is a significant spam phrase if it satisfies equations 5.1 and 5.2.

4. Model selection and Training on User Data: In order to find a suitable model for user profiling, several machine learning and deep learning models were developed, trained and evaluated. The following Machine and deep learning models were analysed - Logistic Regression, Support Vector Machine, Naïve Bayes, Random Forest, Dense Neural Network, and Convolutional Neural Networks. During training, these models were evaluated for different hyperparameters to find the best fitting parameters. The parameter pf is a feature selection parameter and can be used to tune

the size of feature vector space used for the model generation; if its value is less than total phrases found, then all phrases are selected as significant phrases. As the value of pf increases, the less significant phrases are removed from the model.

5. Evaluating models on Test Data: Each of the models developed and trained were evaluated on the test data containing ham and spam samples for performance.

6. Calculating Various Metrics and Plotting Visualizations: During evaluation, to measure the performance of the models for each user, several metrics like accuracy, log loss, confusion matrix, precision, recall, and f1 score were calculated and recorded. Visualisations were plotting confusion matrix, receiver operating characteristic (ROC)/ area under the curve (AUC)curve, and precision recall curve (PRC) explained below.

The classification report represents the classification metrics for each class. The report allows detailed investigation of the performance of the classifier beyond the overall accuracy which can be biased due to inaccuracies in the data. Suryawanshi et al. (2019) used the classification report as a measure of performance evaluation using the metrics precision, recall, F1 score ad ROC-AUC curve. The metrics included in a classification report are presented and discussed below:

Accuracy

This is a metric for evaluating classification models, also known as Accuracy Classification Score (ACS), a measure of closeness of predicted value and actual value which is determined by the total number of predicted outcomes against actual outcomes. ACS measure the total number of predictions made correctly as a proportion of the total predictions made for the test data.

$$Accuracy\ Score = \frac{TP+TN}{TP+TN+FP+FN} x\ 100$$

A higher accuracy score is not a direct determinant for the selection of a model, as in many cases a model with a lower accuracy may have a higher predictive power. This can be illustrated on the case of datasets with large class imbalance where a classification model may achieve a high classification accuracy by correctly predicting the value of the majority class for all the predictions but not for the minority class. Using this model based on an accuracy score would not be useful in the actual domain.

Precision

Precision, also known as positive predicted outcome, is a measure of the exactness of the classification model. It is referred to as measure of closeness of two or more values and means that every time it is measured, the value is the same. It is the fraction of relevant outcomes among the predicted outcomes and is calculated as the number of positive predictions made divided by the total number of positive class labels predicted by the model.

It measures the proportion of positive outcomes predicted correctly.

$$Precision = \frac{TP}{TP+FP} * 100 = \frac{TP}{Total\ Predicted\ Positive} * 100$$

If the precision score is 90%, the classification model will make correct predictions for 90% of all predictions. The precision score is 100% if the classification model produces no FP in the confusion matrix.

Where the cost of False Positives is high, Precision is a good measure to determine. In case of email spam classification, the user might lose important emails if the precision score is low for the spam filtering classification model.

Recall

Recall, also known as sensitivity, is a measure of completeness of the classification model. It is the fraction of relevant predicted outcomes among the total relevant outcomes and is calculated as the number of positive predictions among the total number of positive class labels. It measures the proportion of actual positive outcomes predicted correctly.

Recall calculates the total number of the actual positives the classification model captures by labelling them as positive. This means True Positives among actual positives; hence. recall is also known as true positive rate (TPR).

$$Recall\ or\ TPR = \frac{TP}{TP+FN} * 100 = \frac{TP}{Total\ Actual\ Positive} * 100$$

The higher the value of recall, the lower the number of FN. A low recall indicates many False Negatives.

Recall is a good measure for selecting the best classification model when the cost associated with FN is high. For email spam filtering, a significant number of emails would be received as spam if the recall score is low for the spam classification model which defeats the purpose of the filtering model.

FI score

The F1 score is a function of Precision and Recall, balanced between precision and recall. It is measured as harmonic mean of the two and is calculated as

$$\text{F1 score} = 2 * \frac{precision\ x\ recall}{precision + recall} * 100$$

For a dataset with class imbalance with high occurrences of a class label, there is a need to balance Precision and Recall. In such a case, the F1 score is a good measure as it takes both FP and FN into account. The F1 score shows perfect balance between precision and recall if the value is 1 whereas high imbalance has a value of 0.

AUC/ROC curve

AUC is the area under the ROC curve and ROC is the receiver operating characteristics curve which represents the performance of the classification model for different threshold values and summarizes the capability of the classification models to differentiate between class labels. ROC is the probability curve that is plotted between the true positive rate (recall or sensitivity) on the y-axis and false positive rate on the x-axis of the classification model for different probability threshold values between 0 and 1.

AUC, the area under the ROC curve, represents the entire 2D area under the ROC curve. It shows the aggregate performance of the classification model for all values of the threshold from (0,0) to (1,1) as shown in Figure 5.2.

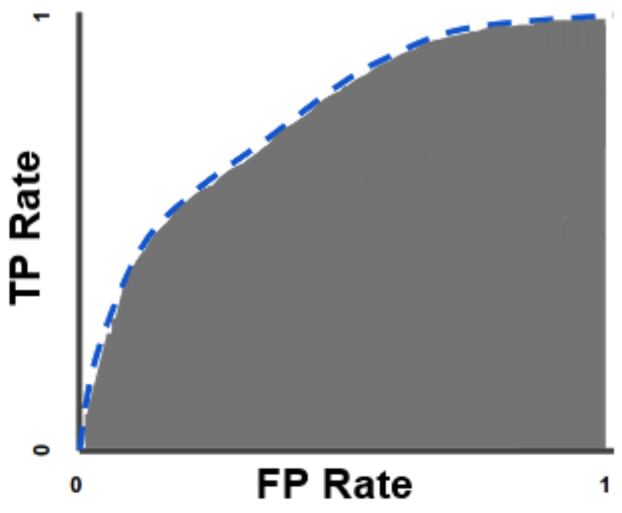


Figure 5.2: AUC ROC curve

AUC can be interpreted as the probability that the classification model will assign a higher rank to a positive than to a negative example, selected randomly. AUC is a scale as well as a threshold invariant.

The value of AUC lies between 0 to 1; the higher the value, the better is the performance of the classification model in classifying spam as spam and legitimate as legitimate. For example, if the AUC value is .96, the model makes 96% correct predictions but if the value is close to 0, This means the model is making high number of incorrect class predictions by classifying spam as legitimate and vice versa.

PRC curve

A precision-recall curve is a trade-off between precision and recall by plotting them as y and x axes for different threshold values between 0 and 1, similar to the ROC curve. PRC measure the performance of the classification model as it evaluates the fractions of true positives amid all positive predictions and hence provides a good measure of future classification performance.

The PRC curve is a straight line at 0.5 which means the dataset contains an equal number of positive and negative instances. The skill of the model is demonstrated by the points above this line. At points (1,1), the models perform efficiently, predicting spam as spam and legitimate as legitimate which means that a curve bowing towards (1,1) is an example of a skilful model. ROC and PRC curves are proportional to each other and have a 1:1 relationship at a certain point.

5.4 Experimental Results

In this section, the details of the experiment and the results are presented. The feature selection approach to develop user profiles was evaluated on the Enron dataset. The Enron has user data for ham and spam emails from 6 users. The dataset size is about 33,716 spam and ham emails for all 6 users. Each of the individual user dataset was divided for training and testing in 70:30 ratios as shown in Table 5.3.

Table 5.3 User datasets for training and testing

User: 1	Train (X, Y): 3666	Test (X, Y): 1506	Train: 70.88 %, Test: 29.12 %
User: 2	Train (X, Y): 4044	Test (X, Y): 1813	Train: 69.05 %, Test: 30.95 %
User: 3	Train (X, Y): 3868	Test (X, Y): 1644	Train: 70.17 %, Test: 29.83 %
User: 4	Train (X, Y): 4214	Test (X, Y): 1786	Train: 70.23 %, Test: 29.77 %
User: 5	Train (X, Y): 3632	Test (X, Y): 1543	Train: 70.18 %, Test: 29.82 %
User: 6	Train (X, Y): 4173	Test (X, Y): 1827	Train: 69.55 %, Test: 30.45 %

(X, Y) represent ham and spam emails in the training and testing samples. The profiles of 6 users were developed from this labelled dataset using the significant phrase approach and evaluated using 3 feature selection methods and multiple classification models. Since the feature set for each user is different, several machine learning and deep learning models were evaluated to determine which one demonstrates superior performance. Each feature selection method was evaluated through two experiments, one with each of the three machine learning classification algorithms, Logistic Regression (LR), Support Vector Machine (SVM), and Naïve Bayes (NB). These algorithms have been detailed in Chapter 4. The second experiment was carried out using the deep learning algorithms Dense nets (Dense Neural Networks, DNN) and Convolution neural networks (CNN) with two different sets of parameters for the size of the input layer.

For evaluation of the performance of models, all metrics, accuracy, precision, recall, and the F1 score were included in the results for these experiments as the inclusion of only one metric may not have provided reliable results, particularly for imbalanced class distributions. Table 5.4 present the class distribution for training and test samples for all user data.

Table 5.4: Class distribution data for all users

Train	Test
User1- Spam 1066, Ham- 2600	User1- Spam 434, Ham- 1072
User2- Spam 1038, Ham- 3006	User2- Spam 458, Ham- 1355
User3- Spam 1028, Ham 2840	User3- Spam 472, Ham 1172
User4- Spam 3154, Ham- 1060	User4- Spam 1346, Ham- 440
User5- Spam 2560, Ham- 1072	User5- Spam 1115, Ham- 428
User6- Spam 3124, Ham- 1049	User6- Spam 1376, Ham- 451

Most research on data evaluations relies on accuracy; however, accuracy is effective for prediction when the values of FP and false FN are very close as was the case for the DMLM and the DMLEM in Chapter 4. For this experiment, however, as can be seen from Table 5.6, the values for FP and FN are somewhat imbalanced and inclusion of other metrics provides better prediction performance evaluation.

Once the profiles for all users were developed, cross verification of each user's preferences for each model with every other user was performed using a random sample of 3,5, and 320 samples from each of the user's test data. These are fed into the trained models for each user and the preferences recorded to note if the user's preferences differ. This is determined by the outcome produced by each model on other users' data.

Table 5.5 reports the results of classification using three machine learning algorithms and the term frequency feature selection method called countvectoriser whereby phrases are converted into vectors to count the occurrences of each in the email training samples. The concept as has been discussed in Chapter 4, Section 4.3 and is designed to count the occurrences of phrases and present them in a matrix with columns as features (significant phrases) and rows as emails.

Table 5.5: Results of classification using term frequency feature selection method

Term Frequency																		
Metric->	Accuracy			Precision			Recall			F1 score			ROC/AUC				PRC	
Model ->	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB
User																		
User 1	97	97	97	99	99	98	97	97	97	98	98	98	100	100	100	99	99	99
User 2	97	97	97	98	97	97	99	99	99	98	98	98	100	100	100	99	99	99
User 3	97	97	98	97	97	98	99	99	100	98	98	99	100	100	100	99	99	100
User 4	97	97	99	98	100	99	90	89	97	94	94	98	99	100	100	100	100	100
User 5	97	97	98	98	98	97	91	93	96	94	95	97	100	100	100	100	100	100
User 6	97	96	98	99	98	94	90	86	96	94	92	95	99	99	100	100	100	100

From Table 5.5, it is evident that on an average the values of all the measures are above 97%, which is a high detection rate. The high value of precision for this method for all three classification models indicates that there is little loss of important emails as FP. High values for recall indicate that a model is efficient in the case of low FN, which indicates low email spam for the user. Recall values are slightly low compared to precision; however, performance is high. The F1 score measures a balance between precision and recall and is of particular concern with imbalanced class distribution; value of >96% on average indicates satisfactory performance.

Table 5.6 FP and FN for each user using term frequency method

Term Frequency						
	FP			FN		
User	LR	SVM	NB	LR	SVM	NB
User 1	36	32	28	10	14	22
User 2	14	8	7	14	42	40
User 3	7	10	4	37	38	28
User 4	44	49	13	9	1	4
User 5	40	29	15	8	10	12
User 6	47	63	17	6	6	26

The number of FP and FN show a similar trend for all classification models with respect to the size of test datasets as shown in Table 5.6. However, there is a difference in the number of FP and FN. Hence, different metrics were considered to evaluate the performance of classifiers using the countvectorising method. Figure 5.3 shows the confusion matrix, ROC/AUC curve and PRC curve for term frequency (countvectorising) method for all six users. The ROC curve is beneficial for evaluating the new tests at an early stage, while the AUC is an effective way of showing overall accuracy which takes the form of a value between 0 and 1. As the value of AUC is approaching 1 for this method, this indicates high accuracy of the outcome from the experiment. Higher values of the PRC curve indicate low FP and FN.

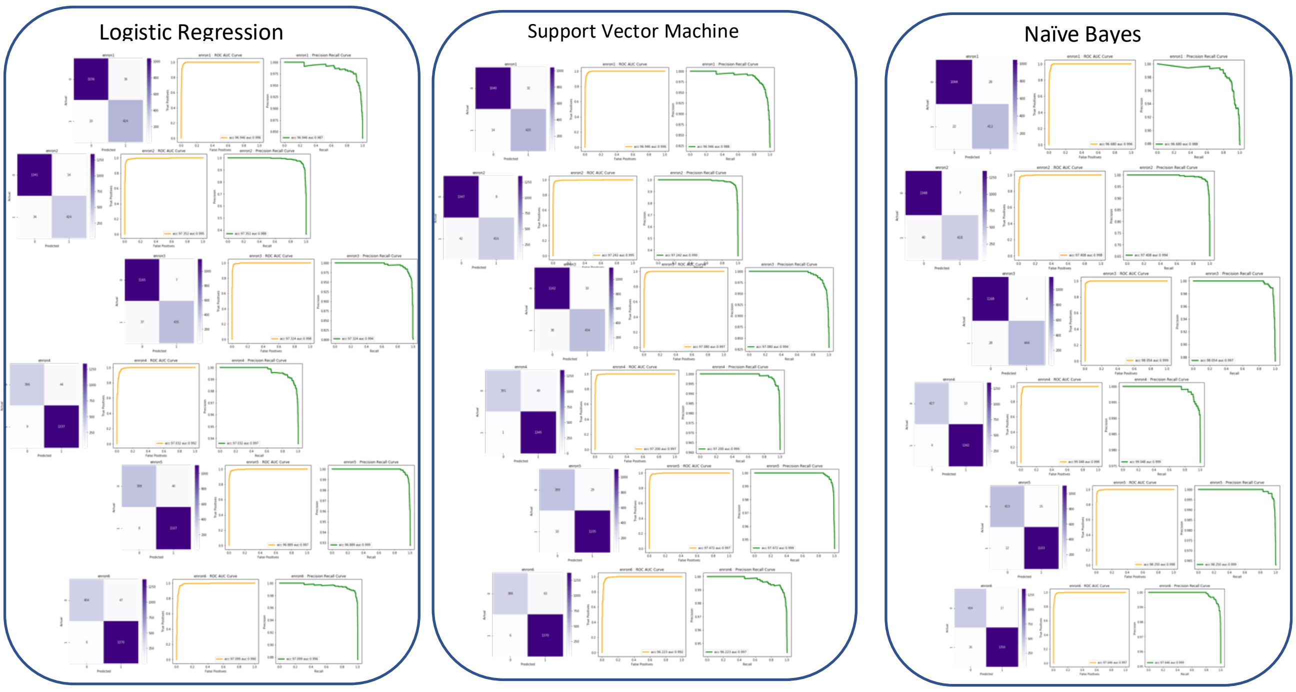


Figure 5.3: Confusion Matrix, ROC/AUC curve and PRC curve for using term frequency method

Table 5.7 shows the results of applying feature selection using tf-idf which reflects the importance of each feature related to each email message in the training set.

Table 5.7: Classification results using TF-IDF

TFIDF vectoriser																		
Metric->	Accuracy			Precision			Recall			F1 score			ROC/AUC			PRC		
Model ->	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB	LR	SVM	NB
User																		
User 1	97	97	100	97	97	97	98	98	92	98	98	94	100	100	100	99	99	99
User 2	97	97	96	97	97	96	99	100	100	98	98	98	100	100	100	99	99	99
User 3	98	98	97	97	97	96	99	100	100	98	98	98	100	100	100	99	100	100
User 4	98	99	99	100	99	100	93	95	95	96	97	97	100	100	100	100	100	100
User 5	98	99	99	98	98	97	96	97	97	97	97	97	100	100	100	100	100	100
User 6	97	97	98	98	97	97	90	92	93	94	95	95	99	99	100	100	100	100

The value of tf-idf increases proportionate to the recurring appearances of the features (significant phrase) in an email message and also in the entire corpus of email datasets. This value is offset by the recurrence of the token in the emails contained in that training corpora, which adjusts for the fact that some words appear more frequently in general in the training email set.

TF-IDF is a numerical statistic which is a product of two statistics namely term frequency and inverse document frequency and is defined as

$$tf - idf(t, d, C) = tf(t, d) * idf(t, C)$$

where *tf(t,d)* is the raw count of the feature t in an email message d which is 1 if the feature exists in an email message and 0 if otherwise; *idf(t,C)* is the amount of information provided by the feature in the email corpus, i.e. it is recurring or not in the entire email corpora. It is calculated by log of inverse fraction of emails that contain that feature obtained by dividing the total number of emails N by the number of emails that contain the token t shown as

$$idf(t,C) = log \frac{N}{d(t)}$$

Hence,

$$tfidf(t,d,C) = tf(t,d) * log \frac{N}{d(t)}$$

The value of tfidf is high if the token frequency in an email is high and low in the entire corpus of emails; the values tend to filter out common features. The ratio of N to the email containing the feature is always >= 1; therefore, the idf value and hence *tfidf* value is >=0. IF the recurring feature is in a great number of emails, the ratio $\frac{N}{d(t)}$ approaches 1, resulting in the *tfidf* to approach 0. This way the number of elements in the matrix reduce iteratively by determining the tfidf and removing the lower contribution features. This whole process is carried out until the matrix contains only the high scoring features. It can be clearly inferred from the performance shown in Table 5.7 that all classification methods for tf-idf have performed comparative to term frequency method.

Accuracy and precision values are high at around 97-98% for all users for all classification methods indicating high detection rates. High values of recall also indicate reduction in email spam. Given the class distribution is somewhat uneven, the value of the F1 score is significant, which is averaging at 96-97%. This means the models are performing well. A high value ROC measures how well the models are classifying ham and spam for each user; from Figure 5.4 AUC values indicate high accuracy of outcomes given by the models for all users.

Figure 5.4 shows the confusion matrix, ROC/AUC and PRC curve, and the high value under the curve indicates high precision and a high recall rate pointing to low values of FP and FN rates.

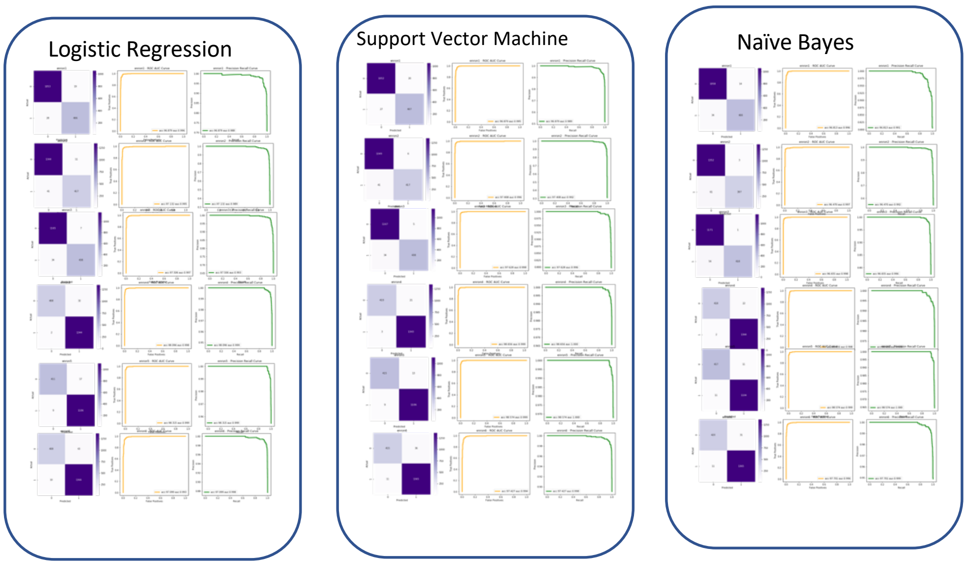


Figure 5.4: Confusion Matrix, ROC/AUC curve and PRC curve for using tf-idf method

The third feature selection method used is word embedding, a feature learning and selection technique where words/phrases from the vocabulary are mapped to vectors analysing its context. Here, glove embeddings (Wikipedia Embeddings (https://nlp.stanford.edu/projects/glove/) have been used for the feature selection. Glove files have been loaded and an embeddings dictionary is maintained which keeps mapping from words to embedding vectors of length 300. Then, for each feature in the mail, it aims to locate embeddings from the embedding dictionary. Vectors of each feature are then normalised to create one vector which represents the selected contents of that email message.

The results for LR and SVM are presented in Table 5.8; the Naïve Bayes is not included in this experiment since embedding produces negative values whereas Naïve Bayes requires all input values to be positive. Hence NB cannot be used for this method.

The overall detection rate using word embeddings is lower for both LR and SVM compared to other methods for all the users. For users 1 and 2 the detection rate with precision and recall and hence F1 score is comparable to other methods. The F1 score at close to 100 indicates low FP and low FN which point to high prediction and low false alarms. However, other metrics have performed at a lower rate.

Table 5.8: Classification Results using Word Embedding

Word Embeddings												
Metric->	Accuracy		Precision		Recall		F1 score		ROC/AUC		PRC	
Model ->	LR	SVM	LR	SVM	LR	SVM	LR	SVM	LR	SVM	LR	SVM
User												
User 1	94.2	95.5	96	97	96	97	96	97	98.4	98.9	96.2	97.3
User 2	94.04	96.1	95	97	97	98	96	97	98	99	95.1	97.7
User 3	93.2	96.2	94	97	96	98	95	97	97.9	99	95.6	97.9
User 4	95.1	96.6	92	95	88	91	90	93	90.5	99	99.4	99.7
User 5	93.5	96.1	89	93	87	93	88	93	97.9	99	99.1	99.5
User 6	93.6	95.7	90	94	83	88	86	91	97.3	98.5	98.9	99.4

As shown in Figure 5.5, the ROC/AUC curve performance is good for all users except user 4 indicating lower accuracy. Hence, the overall accuracy is acceptable for both classification methods though SVM outperforms LR for word embeddings.

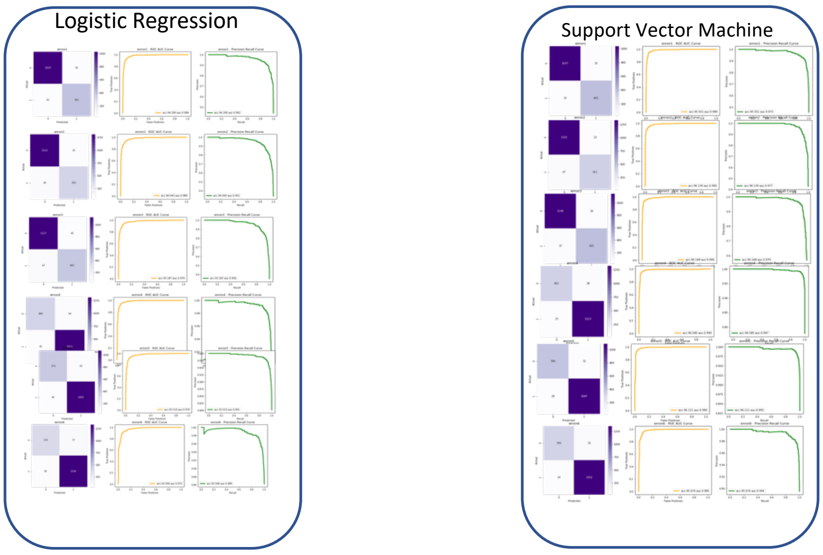


Figure 5.5: Confusion Matrix, ROC/AUC curve and PRC curve for using tf-idf method

Use of Deep learning Algorithms and the Rationale

The next experiment uses deep learning algorithms, Dense Neural Networks also called Dense Nets, and Convolutional Neural Networks with two different input sizes of 300 and 100 referred to as CNN version 1 (CNNv1) and CNN version 2 (CNNv2) in the experimental results. These techniques are explained as follows:

Convolutional Neural Networks (CNN), a type of deep learning algorithm, are specialised types of neural networks introduced in 2012 by Geoffrey Hinton (Krizhevsky, Sutskever, & Hinton, 2012) which are commonly applied to visual data such as audio, images and video in one, two and three dimensions respectively. Lately, CNN have also had significant success with text classification (Bai, Kolter, & Koltun, 2018; P. Wang et al., 2015). CNN convolute several adjacent inputs (known as layer 1) with an activation function applied to each unit which gives an output that forms the next layer (known as output layer 2), and this function is applied to all adjacent input. Next, pooling is performed where the results of output layer 2 are pooled into the next later known as layer 3). This layer consists of fewer units as it averages the results of the units from the previous layer (layer 2). CNN architectures are designed for 2D input data structures, although they can also be used for 1D and 3 D inputs as they offer regularisation of multilayer perceptron's which means, they are fully connected networks where neurons in each layer have a connection with the neurons in the following layer. They consist of several convolutional layers with non-linear activation functions such as ReLU or tanh applied to the output of each convolutional layer followed by fully connected layers. These networks involve two operations, convolution and pooling which leads to feature extraction. Convolution is performed by a convolution layer, a linear operation that involves application of a filter or kernel to an input by multiplying an array of weights to the input data that results in activation. The filter slides over the input multiple times, resulting in a matrix of output maps of activations, a convoluted feature or input feature map, which develops local connections where each region of the input is connected to a neuron in the output. This identifies the features of an input. The value of the filters is automatically learned by CNN depending on the nature of the task and each layer applies different filters which may vary in number and could be hundreds or even thousands. The size of the convolutional features is reduced by pooling layers in order to reduce the computational cost of data processing. Pooling operates on each feature map independently and consists of the application of a certain operation to the feature map to extract a representative output value. Its function is to reduce the spatial size of representation for parameter reduction. It provides a fixed size output matrix that reduces the output dimensions while retaining the salient features that are useful for classification. There are two pooling operations: one is max-pooling that selects maximum value in each selected region of the map and the second is average-pooling that selects the average value resulting in a singular scalar output which significantly reduces the output size. The features identified from all convolutional and pooling layers are passed on to the last fully connected layer, e.g. a multilayer perceptron with dropout and softmax output that uses high-level features for development of the training model to be used for classification.

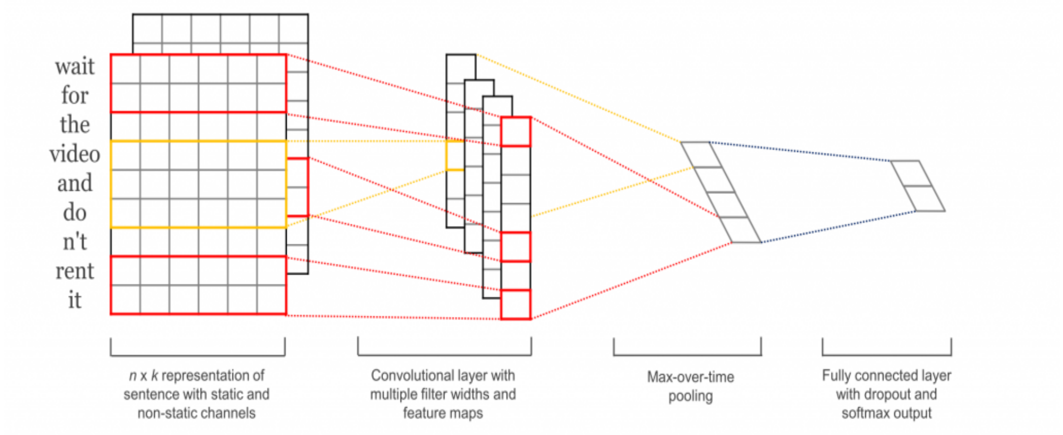

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 5.6: Layers of a Convolutional Neural Network (CNN) for text classification

While applying CNN in Natural Language Processing for text classification problems, text is represented as a 1D array as shown in Figure 5.6. This means the CNN architecture consists of 1D convolutions and max pooling (Jacovi, Shalom, & Goldberg, 2018). It considers words as n-grams which is used to classify a document to a predefined category as in the case of email spam filtering where each email is classified as spam or legitimate. Filters with width usually of the size of an input matrix slide over the rows of the matrix of words to develop feature maps; the height of the filter or region of the input matrix covered by the filter may vary.

Given a set of inputs x_i and outputs as y_i belonging to a one-dimensional convolutional layer where N is a neuron (though it can be multiple neurons as described above), then, output is

$$y_i = N(x_{i,}\ x_{i+1,}x_{i+2}, \ldots.)$$

and Neuron N is described as

$$\sigma(w_0x_{0,}\ w_1x_{1,}w_2x_2, \ldots.)$$

with weights $w_0, w_1, w_2, \ldots$ and activation function σ that control the behaviour of neurons and may hold positive or negative values. Neurons with the same weight are identical and are handled by convolution. All neurons in a layer are described at once. If W is the weight matrix then

$$W = \begin{bmatrix} W_{0,0} & W_{0,1} & W_{0,2} & \ldots \\ W_{1,0} & W_{1,1} & W_{1,2} & \ldots \\ W_{2,0} & W_{2,1} & W_{2,2} & \ldots \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix}$$

and

$$y_0 = \sigma(W_{0,0}x_{0,}\ W_{0,1}x_{1,}W_{0,2}x_2, \ldots.)$$
$$y_1 = \sigma(W_{1,0}x_{0,}\ W_{1,1}x_{1,}W_{1,2}x_2, \ldots.)$$
$$y_2 = \sigma(W_{2,0}x_{0,}\ W_{2,1}x_{1,}W_{2,2}x_2, \ldots.)$$

Though the weight matrix connects every input to every neuron with varying weights, it is possible that the same weights appear multiple times and neurons may not connect to all inputs. In that case, the weight matrix may change with some values as zero.

The pooling layer uses max or average operation to combine the output of all the convolution windows into a vector of predefined dimension. This output vector is then fed to the follow-on layers and finally into the fully connected layer for classification. An example of CNN of text classification (Ye Zhang & Wallace, 2015) is shown in Figure 5.7.

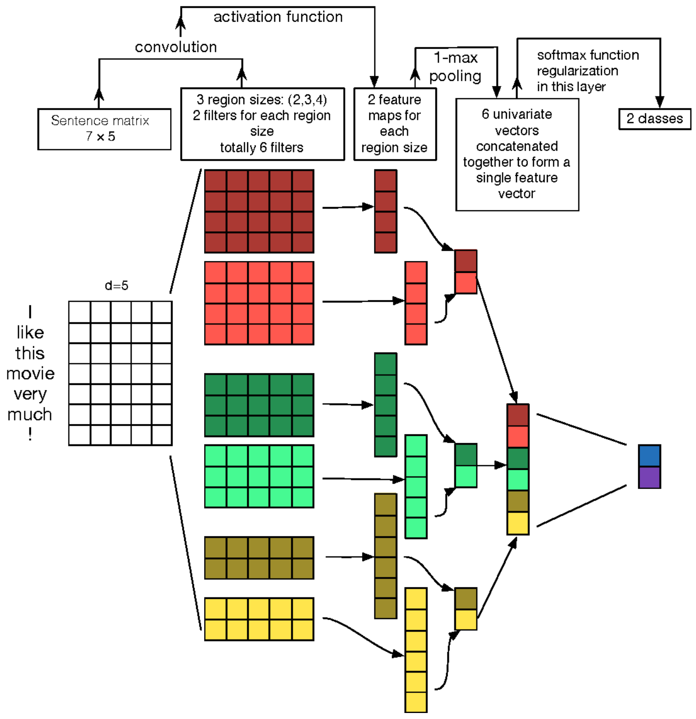

activation function
convolution
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
1-max pooling
6 univariate vectors concatenated together to form a single feature vector
softmax function regularization in this layer
2 classes
d=5
I
like
this
movie
very
much
!

Figure 5.7: Architecture of a CNN for text classification.

Here, the input matrix has dimensions of size five with three filter regions of sizes 2, 3 and 4 with two filters each making a total of six filters. Each filter slides over the sentence matrix to perform convolution and generates two feature maps for each region. The feature maps vary in size for each of the three regions. Each of the 6 feature maps is applied with 1-max pooling in order to obtain the largest number from each feature map to generate 6 univariate feature vectors. These vectors are concatenated to form a single feature vector to be fed into the final layer to be used for binary classification with 2 output labels.

There is no evidence of use of CNN for user profiling though it has been used for text and sentence classification in the following examples. P. Wang et al. (2015) proposed a hierarchical semantic model using clustering and CNN for classification of short texts. This novel method used a fast clustering algorithm for introducing additional knowledge using word embeddings which was followed by detection of multi-scale sematic units (SU), fed into the CNN model as features. The model was tested and validated for effectiveness on text snippets obtained from Google snippets and TREC datasets. Jacovi, Sar Shalom, et al. (2018) also worked with details for applying CNN to processing of text for classification. The authors carried out a hypothesis examination that global max-pooling support filters in detection of n-grams in CNN and showed that using distinct activation patterns, filters may capture multiple semantic classes of n-grams. Important n-grams can be separated by global max-pooling. Hence, filters are not homogenous and capture linguistic patterns which can be detected by clustering high scoring n-grams according to their activation patterns. Based on this paper, it is evident that model-based as well as prediction-based interpretability of CNN can be improved. Ye Zhang and Wallace (2016) conducted a sensitivity analysis to determine the effects of hyperparameters for model architecture of CNN on the performance of the model. The baseline configuration used was Google word2vec word vectors, region size of 3,4,5, with 100 feature maps, and ReLU activation function, 1-max pooling, and a dropout rate of 0.5. The effect of this configuration on nine selected datasets was observed and the authors concluded that hyperparameters can be effectively used with CNN for sentence classification.

Dense Neural Networks (DNN) is a supervised deep learning algorithm that develops neural networks where the layers are fully connected by the neurons. Each individual neuron in a layer receives input from all neurons in the preceding layer causing the layers to be densely connected (hence the name DNN). Dense layer is another name for fully connected layers where all neurons in a layer are fully connected to all neurons in the following layer. Dense layers are also referred to as hidden layers and use a ReLu activation function. Thus, a dense layer is a linear operation where every input is connected to every output by weights and is generally followed by a non-linear activation function. For binary classification, sigmoid is used as final activation function for the output layer. Output, O of the dense layer with input X is implemented as

$$O = \sigma(dot(X, W) + b)$$

Where σ, the activation function is ReLU for each of the hidden layers and sigmoid for the output layer, W is the weight matrix created by the layer and b is the bias vector created by the layer.

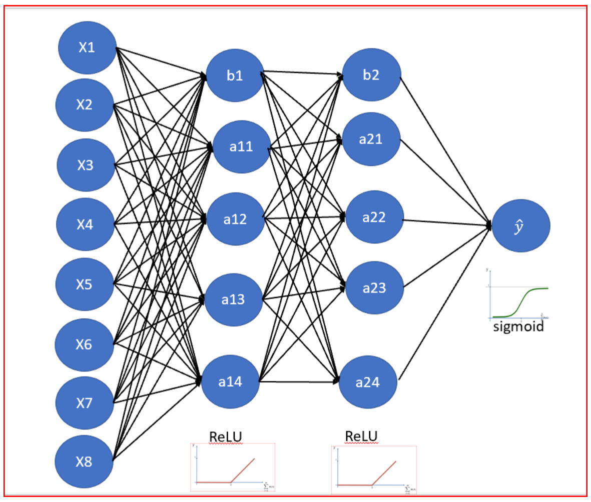


Figure 5.8: Fully connected layers of Dense Neural Network

After defining the layers, the neural network is compiled and the loss function between actual and predicted output is calculated. The performance of the dense neural network is evaluated using metrics such as accuracy, precision, recall or confusion matrix.

The layers in DNN provide features from all combinations of features from the preceding and subsequent layers for learning whereas CNN relies on consistent features. DNN has been used for image and audio classification whereas their applications for text classification is limited.

Volz et al. (2016) conducted a comparative study of performance of data-driven architectures based on SVM and dense neural networks for time series classification of the intention of pedestrian to cross the street. The model was evaluated for several time horizons and result demonstrated promising accuracy for pedestrians' intent of crossing. Even long-time, the accuracy of the DNN proposed in this architecture is consistently above 80% for the time horizon of 6 secs.

Brun, Yin, and Gelenbe (2018) presented a methodology using dense neural networks for detection of network attacks in an online setup. The proposed methodology detected the attacks against IoT gateways from the packet capture metrics by predicting the probability of an attack. For evaluation of the method, the attacks were inserted into packet captures and results show correct detection of attacks by DNN. Further evaluation of the model using the same dataset using a threshold showed comparable accuracy values.

M. Du et al. (2019) took a novel network structure approach based on fully dense neural networks (DNN) to reduce the pre-processing difficulty and high memory consumption associated with CNN. The proposed DNN model performs feature extraction using network blocks and reduces the model size with dense connect. The evaluation results show that by using dense connections, features are extracted efficiently while reducing the model's parameters and delivering high recognition rates.

The performance of the three classification models is presented in Table 5.9 where DNN outperforms both versions of CNN in terms of the overall detection rate as well as precision and recall, indicating low FP and FN rates.

Table 5.9: Classification results using deep learning models

Deep Learning																		
Metric->	Accuracy			Precision			Recall			F1 score			ROC/AUC			PRC		
Model ->	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2
User																		
User 1	96.48	93.7	88.6	99	97	94	96	94	89	97	96	92	0.987	97.9	89.4	0.969	93.6	76.2
User 2	97.5	94.8	92.1	98	97	95	99	96	94	98	97	95	99.3	94.8	92.2	98.8	95.1	79.2
User 3	97.32	95.5	71.2	97	96	71	99	98	100	98	97	83	99.5	98.9	50	99.2	97.2	64.4
User 4	96.36	75.34	92.2	100	75	98	85	100	70	92	86	82	97	50	92.5	99	87.7	96
User 5	96.89	72.26	91.25	97	72	95	91	100	73	94	84	82	99.6	50	94.1	99.8	86.1	96.3
User 6	95.13	88.7	75.3	97	96	75	83	57	100	89	71	86	95.7	90.5	50	98.5	95	87.7

The ROC/AUC and PRC curves show a similar trend where DN performs better than CNN.

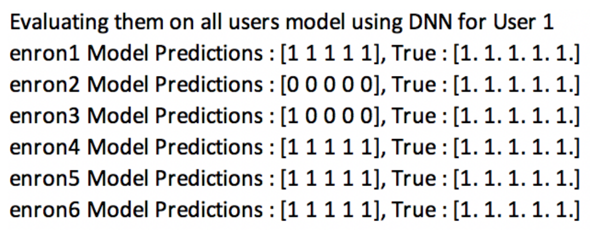
5.5 Cross Verification of User Preferences

In this section, a cross check of user preferences was performed to determine the applicability of a classification model developed for one user for the data for another. Since the feature set for each user is individual and different to another user in principle, this experiment is conducted to see how different the classification outcome for several users would be. This is to check if spam mail for one user might be ham for another user.

This done by selecting random samples of data from user test data samples provided as input to the classification model for prediction (for user 1). Subsequently, the same samples were fed into classification models developed for 5 other users and the outcome recorded. This experiment was conducted using deep learning methods and outcomes were compared with the true values of those test samples. The configuration chosen were 3, 5 and 20 random samples from the test samples for each user.

The experimental results indicated that sample size of 3 was too small and did not produce any deterministic results. Hence, this sample size was discarded. For the sample sizes 5 and 20, predictions made by the classification model for all 5 users are compared against the true values of those test samples. The value '0' means ham and '1' indicates spam in the results tables. Table 5.10 shows the evaluation results for the 5 test samples belonging to User 1 provided as input to all 5 users to individual DNN models for classification. It shows that the 'True' label was the same as the prediction label for users 1, 4, 5 and 6; however, the prediction is different for user 2 and 3.

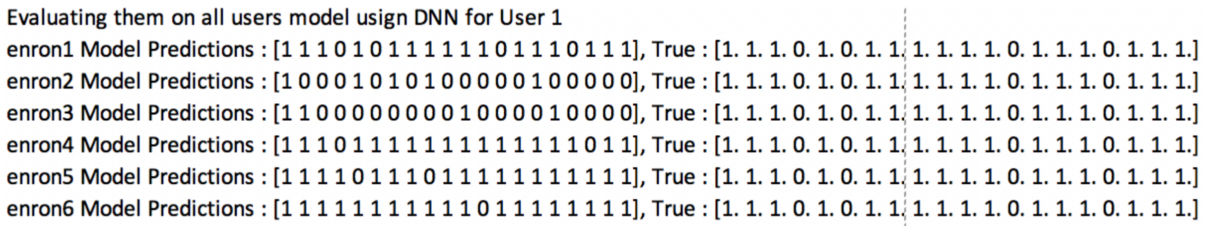
Table 5.10: Comparison of user preference of User 1 with other users using 5 samples



```
Evaluating them on all users model using DNN for User 1
enron1 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron2 Model Predictions : [0 0 0 0 0], True : [1. 1. 1. 1. 1.]
enron3 Model Predictions : [1 0 0 0 0], True : [1. 1. 1. 1. 1.]
enron4 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron5 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron6 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
```

Table 5.11 reports results for the same user and 20 user data samples. Here, the DNN model for each user loads 20 randomly taken spam emails from the test data of User1 and then evaluates these using the DNN model for all users. The prediction made by the model showed that the DNN performance for User 1 was the same as the true value for those samples. The predictions made by the models for other users recorded the difference in preferences for different users as their model predicted the same samples differently.

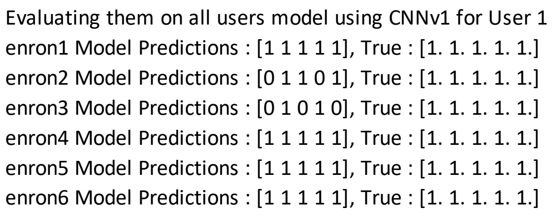
Table 5.11: User preference Comparison of User 1 with other users using 20 samples with DNN model



```
Evaluating them on all users model usign DNN for User 1
enron1 Model Predictions : [1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1], True : [1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1.]
enron2 Model Predictions : [1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0], True : [1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1.]
enron3 Model Predictions : [1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0], True : [1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1.]
enron4 Model Predictions : [1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1], True : [1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1.]
enron5 Model Predictions : [1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1], True : [1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1.]
enron6 Model Predictions : [1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1], True : [1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1.]
```

Tables 5.12 and 5.13 show the results for the same User1 sample test data of 5 and 20 using CNN version 1. The prediction results by the CNNv1 model for users indicate that similarity in preferences for users 1, 4, 5 and 6; however, they are different for users 2 and 3.

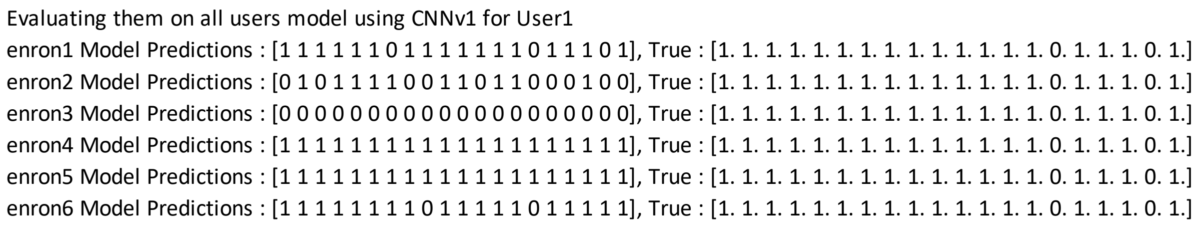
Table 5.12: Comparison of user preference using 5 samples with CNNv1



```
Evaluating them on all users model using CNNv1 for User 1
enron1 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron2 Model Predictions : [0 1 1 0 1], True : [1. 1. 1. 1. 1.]
enron3 Model Predictions : [0 1 0 1 0], True : [1. 1. 1. 1. 1.]
enron4 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron5 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron6 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
```

The prediction results for 20 samples from all 6 users using CNNv1 indicate that user 1 does have some level of similarity of preference to users 4, 5 and 6, but variations are visible when the sample size is larger. There is about 50% commonality in prediction between user 1 and 2 from this sample of 20.

Table 5.13: Comparison of user preference using 20 samples with CNNv1



```
Evaluating them on all users model using CNNv1 for User1
enron1 Model Predictions : [1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1], True : [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1.]
enron2 Model Predictions : [0 1 0 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 0], True : [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1.]
enron3 Model Predictions : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0], True : [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1.]
enron4 Model Predictions : [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1], True : [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1.]
enron5 Model Predictions : [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1], True : [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1.]
enron6 Model Predictions : [1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1], True : [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1.]
```

For the same sample sizes for User 1 tested through CNN version 2 (shown in Table 6.14), the prediction made by the classifier and true values vary for User1, which signifies a low detection rate performance by CNNv2. This performance of CNNv2 is reflected in experiments while developing user profiles as well. The preferences of users 4, 5, and 6 were predicted to be the same as the true value of the samples but varied for users 2 and 3.

Table 5.14: Comparison of user preference using 5 samples with CNNv2

```
Evaluating them on all users model using CNNv2 for User 1
enron1 Model Predictions : [1 1 1 1 0], True : [1. 1. 1. 1. 1.]
enron2 Model Predictions : [0 0 1 0 1], True : [1. 1. 1. 1. 1.]
enron3 Model Predictions : [0 0 0 0 0], True : [1. 1. 1. 1. 1.]
enron4 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron5 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
enron6 Model Predictions : [1 1 1 1 1], True : [1. 1. 1. 1. 1.]
```

Contrary to low detection performance for sample size 5, CNNv2 performed accurately for sample size 20 where predictions made by the model were the same as the true value of the samples for User1 as shown in Table 5.15. The preferences indicated by this model for all other 5 users are different to user 1 as prediction value and true values are different.

Table 5.15: Comparison of user preference using 20 samples with CNNv2

```
Evaluating them on all users model using CNNv2 for User 1
enron1 Model Predictions : [1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0], True : [1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.]
enron2 Model Predictions : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0], True : [1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.]
enron3 Model Predictions : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0], True : [1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.]
enron4 Model Predictions : [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1], True : [1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.]
enron5 Model Predictions : [1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1], True : [1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.]
enron6 Model Predictions : [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1], True : [1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.]
```

Similarly, comparative evaluation of user preferences for users 2-6 indicates patterns that show the difference in user taste for emails.

5.6 Analysis of Results and Discussion:

Experiments have been carried out using three feature selection methods on the significant phrases selected from the datasets for the 6 users from the Enron dataset using machine learning and deep learning algorithms to determine which model to select for user profiling.

From the results, it transpires that though performance is comparable, the Naïve Bayes classification algorithm performs better than LR and SVM for term frequency and tf-idf methods. For word embeddings, SVM performs better than LR.

For deep learning methods, DNN performs better than any of the selected versions of CNN. Hence, the recommended model with significant phrases as feature for developing user profiling can be implemented with any of the machine learning or deep learning models suggested.

For cross verification of user preferences, the experiment compared the prediction values for the samples for all 6 users to identify the variation in user preferences in terms of receiving emails in their inbox. The same email is classified spam to one and ham to another. Sample sizes of 5 and 20 emails were randomly selected from the test data samples and given as input to classification models of DNN, CNNv1 and CNNv2 for training and then evaluation. The evaluation results show a similar pattern for user preferences for all 6 users for DNN and CNNv1 for both sample sizes. This indicates for all three models using sample size of 5 that there are similar preferences for users 1, 4, 5 and 6; however, for large samples of 20, some differences could be discerned among these users. The results for CNNv2 show difference in predicted and true values for users 2 to 6 for 20 samples indicating a different pattern as compared to DNN and CNNv1. Tables 5.10-5.15 show the experimental results for User 1, similar comparisons were performed for all other users to find the differences in predicted and true values for the same email due to differing user preferences.

Hence, it can be deduced that user preferences differ and do play an important role in email spam classification and detection.

Chapter 6

Hunting Random Strings and Topic-based Detection Methods

Spam filters are supposed to block e-mail scams from ever reaching us, but criminals have learned to circumvent them by personalizing their notes with information gleaned from the Internet and by grooming victims over time.
-Maria Konnikova

6.1 Overview

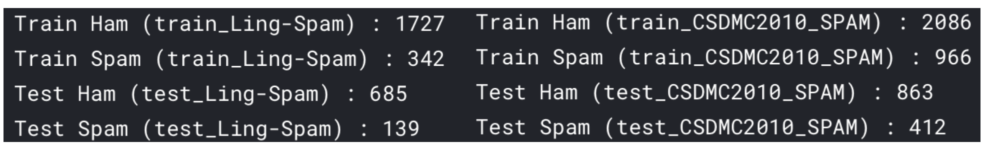
The previous two chapters focused on techniques for increasing the performance of email spam classification at the client side. A dynamic multi-layer model (DMLM) using a novel technique for feature selection filters was proposed in Chapter 4 and evaluated using CART machine learning ensemble of decision trees. The results showed high detection rates for email spam. An enhancement of this model was proposed in Chapter 5 where a bagging ensemble of classifiers was proposed. The models were evaluated and validated for performance with five different datasets. Preference of individual users differ; this was demonstrated in Chapter 6. Through experiments and results it was seen that email spam detection improves when performed by profiling users. Experimental results also showed that user tastes differ, and the same email may be classified differently by different users.

It has been established that spammers continue to introduce new techniques for deceiving the efforts of email spam filtering mechanisms. One of the latest techniques involves injection of good text into email spam. The purpose of such injection is to confuse the filters since most of the feature selection techniques rely of the frequency of words. Hence, the performance of content-based email spam filters deteriorates. Phishing is one type of email spam that leads recipients of email to click on a link and compromise their identity; if successful, this may lead to financial and non-financial losses to users. An injection of random good text into spam emails also impacts the capability of phishing email filters. Hence, it has a wide range of impacts. There is no evidence from research to determine the impact of random text injection on classification filters. This chapter proposes to fill this gap through an entropy method for random string detection to identify email spam based on experiments where random good text is first injected into sample spam email datasets followed by an application of the proposed entropy method.

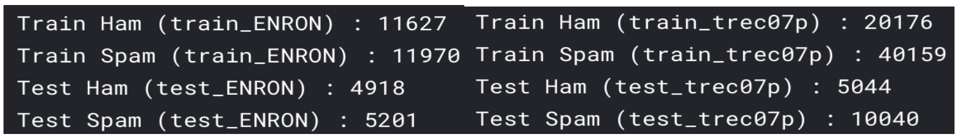
Email spam has been addressed in this thesis and in other existing literature whereby most of the features for spam filtering are extracted from either the body or header of the email messages or the entire email message content. Little attention had been given to the topic/subject of the email spam. Topics of emails are crafted to lure users to open, read and react to the emails. Most of the email spam filters focus on the body of the email content and only a small portion consider the subject field for classification. So far, this research has proposed techniques based on the entire email, per se header, subject and body of the email and no close attention has been paid to the subject field. This chapter has one set of experiments focused just on subject field incorporating the topic of the emails as a feature into the email spam classification framework. Feature selection is carried out using GloVe word embeddings and a range of machine and deep learning models were developed for classification and their performance measured and evaluated. These proposed models are validated using four datasets with sample sizes ranging from approx. 3000 to 75000 emails. The results were recorded and compared for each of the proposed classification models.

In this chapter, four datasets - TREC07, Enron CSDMC2010, and Ling-spam, of varying sizes were used for training and evaluation purposes. Each dataset was divided randomly in a ratio of 70:30; 70% was used for training and 30% for testing. Due to the shortcomings of the holdout method, where the emails may be unevenly divided as per the split percentage k-fold, cross validation was used in the experiments where the entire dataset was divided into k times. With a 70:30 split, the model was trained with 70% of the data and tested on 30% - performed k times with for each operation.

The dataset sizes for each were as follows, TREC07 (75419), Enron (33716), CSDMC2010 (4327), Ling-spam (2893). The breakdown of ham and spam divided into training and testing at a 70:30 ratio is shown is Figure 6.1.



```
Train Ham (train_Ling-Spam) : 1727  Train Ham (train_CSDMC2010_SPAM) : 2086
Train Spam (train_Ling-Spam) : 342  Train Spam (train_CSDMC2010_SPAM) : 966
Test Ham (test_Ling-Spam) : 685     Test Ham (test_CSDMC2010_SPAM) : 863
Test Spam (test_Ling-Spam) : 139    Test Spam (test_CSDMC2010_SPAM) : 412
```



```
Train Ham (train_ENRON) : 11627  Train Ham (train_trec07p) : 20176
Train Spam (train_ENRON) : 11970 Train Spam (train_trec07p) : 40159
Test Ham (test_ENRON) : 4918     Test Ham (test_trec07p) : 5044
Test Spam (test_ENRON) : 5201    Test Spam (test_trec07p) : 10040
```

Figure 6.1: Datasets with their ham/spam training and test split

The datasets are described below in ascending order of size; the original dataset size is used for experiments in this chapter:

TREC07: This email dataset consists of total 75,419 messages out of which 25,220 are legitimate emails and 50199 are spam messages.

Enron: The labelled email dataset size is 33,716 messages with 16,545 ham and 17,171 spam email messages.

CSDMC2010: This dataset has 4327 labelled messages where 2949 are legitimate and 1378 are spam email messages.

Ling-spam: The dataset consists of a total of 2893 email messages, of which 2412 are legitimate and 481 are spam messages.

Performance Evaluation Metrics

For classification, there are two types of outputs generated: class output and probability output. For binary classification, such as email spam detection, the class outputs can be either 1(spam) or 0(ham). SVM produces a class output. Logistic Regression, Naïve Bayes, Random Forest and XGBoost produce probability outputs which can be converted into classes by defining threshold probability values for the output classes. This chapter uses seven metrices – confusion, accuracy, precision, recall, F1 score, ROC/AUC curve and PRC curve to evaluate the machine learning classification algorithms used for email spam detection. Chapters 4 and 5 have description of all metrices except the confusion matrix which is described a below:

Confusion matrix-

A confusion matrix records predicted results versus actual values in an n by n matrix where n = the number of predicted classes. This explicit arrangement provides visual representation of the performance of a classification model on a specific test data; one side of the matrix displays the predicted results, and the other side displays the actual data provided the actual values are known. Table 6.1 presents a sample confusion matrix for binary class outputs (negative and positive). The predicted labels are presented horizontally, and the actual labels are shown along the side of the matrix. Each of the cells holds the total number of predictions made by the classifier. The cell values in the matrix are 'True' if the actual is as predicted and 'False' if is not. A confusion matrix is an effective measure to determine the classifier's capability to make correct predictions.

For binary classification problems as is the case for email spam, classification with class outputs as spam and ham, n = 2, the matrix is of the size 2 x 2. If spam = positive and ham = negative then True negative means the email message is ham and the classifier correctly predicted it as ham, True positive means the email message is spam and the classifier correctly predicted it as spam. Incorrect classification made by the classifier model are recorded as False positive and False negative. M. Iqbal et al. (2011) used a confusion matrix to evaluate the performance of the Naive Bayes classifier against the k-NN classifier for email spam filtering.

Table 6.1 Confusion Matrix

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Visualisation of the performance of the classifier model through the confusion matrix shows that the class distribution for the actual test data counts and the classifiers' predicted data counts for each of the cells. The confusion matrix is also known as error matrix where the cells of Table 6.1 represent the breakdown of error types as true positive (TP), true negative (TN), false positive (FP) and false negative (FN) defined as

FP- False positive means that a legitimate email which is negative in actual data, has been incorrectly predicted as spam email by the classifier model
FN- False negative means that an email spam which is positive in actual data has been incorrectly predicted as legitimate by the classifier model
TP- True positive means that an email spam which is positive in actual data, has been correctly predicted as spam by the classifier model
TN- True positive means that a legitimate email which is negative has been correctly predicted as legitimate by the classifier model

The chapter is structured as follows: Section 6.2 introduces methodology for the proposed method of email spam detection for the case of random string injection in email spam, Section 6.3 reports experimental results and analysis followed by discussion in Section 6.4. Topic-based classification is described in Section 6.5 along with experimental results and analysis. The final Section 6.7 is the conclusion of the chapter.

6.2 Random String Injection- Using Entropy for Classification at Client Side

Statistics-based techniques have been effective for email spam filtering; however, spammers constantly find techniques to make these filters less effective by modifying the content of email spam. They identify the probabilities associated with spam words and either obfuscate them or introduce words with ham probability to reduce the 'spammy' character of the emails. This technique is called 'good word attacks' where spammers inject good words into email spam (Y. Zhou, Jorgensen, & Inge, 2007). Some research has gone into finding solutions to this problem (Xiao-wei & Zhong-feng, 2012; Yan Zhou, Jorgensen, & Inge, 2008). Lowd and Meek (2005) found that the Naïve Bayes filter was extremely vulnerable to attacks where spammers inserted words indicative of ham emails into spam emails and suggested re-training the filter to minimise the effectiveness of such attacks. However, none of these solutions address if the problem is transformed from injection of good words to spammers injecting random pieces of valid text rather than just words in the spam email to confuse the filters. Such text has low spam probability hence adding it to email spam reduce the overall spam probability of the message. This makes signature-based techniques ineffective and also changes the digital signature of an email. Pelletier et al. (2004) referred to the addition of random text but did not address the issue or suggest a solution. Furthermore, so far there has been no evidence of research into this problem.

In this work, this problem is first validated and then a solution suggested. To validate the problem, two sets of experiments were conducted. First, random valid pieces of text representing 20% of the size of an email was injected into email spam datasets. Secondly, this same 20% text was scrambled before injection. The entropy measure was then used to identify if this injection introduces adversity in email spam classification.

Accuracy and precision of segregating ham and spam emails using machine learning algorithms is heavily dependent on the features selected for classification. These mathematical models lack a deep understanding of the problem domain and rely on parameters estimated form the training samples. Chapters 3-4 have suggested techniques for improving feature sets to improve the performance of email spam detection methods. This chapter proposes another method to further improve the technique to detect and classify spam and ham. A machine learning model may perform well on one dataset, but if the characteristics of another dataset are significantly different, the same model may perform badly. To maximise the email spam detection rate, it is proposed to use an entropy method which brings statistical and probability theories together for decision making. To measure the effectiveness of the proposed approach, combinational tests were performed with machine and deep learning algorithms on several datasets listed in earlier sections.

6.2.1: Methodology

The methodology used in this experiment involves the following steps

- **Data Pre-processing**

Here data sample are taken from all datasets and below mentioned steps are performed

- Loops through each file
- Tokenize file contents ignoring unwanted characters using ISO-8859-1 encoding
- Remove stop words which are commonly occurring words in a sentence and does not contribute much to classification.
- Reconstruct text contents after all above steps have cleaned the data samples.

- **Feature selection**

Feature extraction and selection is performed with the following methods: Every email is represented as a word vector of dimension N where each dimension of the vector relates to a certain weight of the term resulting in a Vector Space Model. This weight can be the term frequency. The sample data is represented in a sparse matrix of dimension M x N where M is the number of emails in a sample set.

- **Model building and parameter selection**

Once feature selection is performed, this step involves building models and selection of optimum hyper parameters to train it for classification. A grid search through each of the models is performed to find the best parameters.

There are classification algorithms available that are suitable to be used as models for email spam filtering. The sheer volume of available machine learning algorithms is overwhelming, and it is important to know the ones suitable for the task in-hand based on research. In previous chapters, the following machine learning algorithms - CART, NB, SVM, kNN, LR and deep learning algorithms - DNN and CNN have been described and used for classification.

This section discusses the classification algorithms presented in this chapter which are used in the proposed framework for this research. Each of the algorithms can be matched with the most appropriate classifier and can learn to perform the classification task. Instance representation is required by all machine learning algorithms for training and evaluation. Email spam is an instance which is transformed into a vector ($x_1, \dots, x_m$), where $x_1, \dots, x_m$ are attribute values for X1, . . . ,Xm, similar to the vector space model in information retrieval (I. Androutsopoulos et al., 2000). The reason for this transformation is the limitation of most machine learning algorithms, namely that they can only classify numerical objects like vectors.

This chapter uses the following machine learning algorithms: Logistic Regression, Support Vector Machine, Naïve Bayes, Random Forest and Extreme Gradient Boosting, and deep learning (CNN and DNN) to build models for classification. A brief description of the layers of DNN and CNN is provided later in this section.

The arguments for the choice of these algorithms, and example instances where they have proven effective for classification, are discussed as follows: Naïve Bayes is the simplest and most widely used algorithm for classification. Studies published over the last two decades that carried out a comprehensive comparison of Naïve Bayes with other classification methods, show that Naïve Bayes classification is outperformed by more recent approaches (Chakrabarty & Roy, 2014). However, Naïve Bayes requires only small amounts of data for parameter estimation and, due to the assumption of independent variables, variable variance for each class is to be determined as variation in each variable which contributes to variation in each class. Hence, Naïve Bayes is included in the classification method of this chapter.

Support vector machines (SVM) have proven to be more effective than the benchmarks of effectiveness set by machine learning algorithms and hence are the method of choice for data scientists as they are considered advanced machine learning algorithms for binary classification such as email spam detection. The machine learning algorithms namely, Naïve Bayes, and Support vector machine are being used for classification in several datasets as they provide quality classification results (Youn & McLeod, 2007). A comparison of performance of NB and SVM algorithms is presented in Harisinghaney et al. (2014) where both algorithms show acceptable accuracy for Enron dataset. Logistic regression is useful for reducing noise in the training data before preparing for classification (Wijaya & Bisri, 2016; Yang, Liu, Zhou, and Luo 2019), is easy to implement and efficient in terms of training. XGBoost implements optimisation of gradient boosting trees as an ensemble and has many strengths such as regularisation and parallelisation that reduces overfitting, has high speed, and handles missing values. These attributes and its evaluation in terms of accuracy makes it a preferred choice in data science and machine learning; however, its ensemble structure can increase the computation time. Random forest as another ensemble of decision trees shows good performance for accuracy and computation time. A study by Hajara et al. (2019), trained, tested and evaluated XGBoost and RF using the same metrics on the same datasets for phishing detection. XGBoost proved to be robust and outperformed RF for some of the problems whereas RF handled overfitting better than XGBoost, reduced variance and is a collection of independent classifiers as it is an example of a bagging ensemble of decisions trees. Hence, these algorithms are selected for use in the model for classification.

The deep learning algorithms have been described in detail in Chapter 5, the argument for their choice here is explained as follows: Dense Neural Networks (DNN) consist of significant numbers of linear neural net layers - models earlier known a Neural Networks (now DNN).

Convolutional Neural Networks (CNN) are deep forward feeding artificial neural networks where the connections between nodes are not cyclic. Though DNN and CNN are conceptually and intuitively applied for image classification, their quick model development and their high speed is an advantage also for text classification as messages contain large numbers of features although M. Du et al. (2019) have argued that DNN proved more efficient than CNN for their case study. Ho, Baharim, Abdulsalam, and Alias (2017) carried out a review of deep learning methods and their application for text categorisation. The authors discussed and compared dense verses convolutional layers in neural networks. For DNN, the input neurons are fully connected to the neurons in the next layer whereas in CNN, convolution is performed on the input layer where multiple filters, used for text categorisation, glide over each row of the matrix containing layers between the input and output layers from which the output is calculated. As DNN are distinct from CNN in terms of fully connected layers and application in natural language processing, both methods are chosen as deep learning models for email spam filtering in this thesis.

Random Forest and Extreme Gradient Boosting are two machine learning algorithms that are introduced in this chapter. The description of these algorithms and the example instances where they have proven effective are presented as follows:

Random forest (RF), a non-parametric supervised machine learning algorithm proposed by Breiman, (2001), is a combination of a large number of individual decision trees – a forest - which can be used for classification and regression and to build a model with reliable prediction outcomes. It is based on the bagging method that collective learning improves the outcome. The model collects output from each individual tree and uses a voting mechanism as aggregation of votes to decide the final outcome (Goel, 2017). As the size of the trees grow, the algorithm searches for the best features among those collected. However, this does not increase complexity as hyperparameters have few requirements. Each tree in RF is trained with a subset of the training data and depends on the independently selected random vector with the same distribution for all trees within the forest (Figure 6.2).

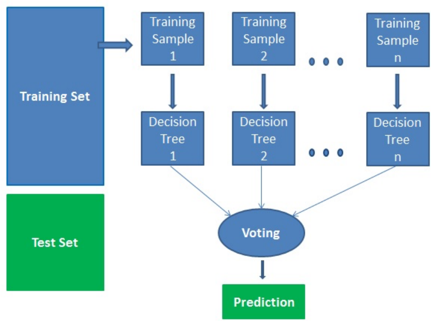


Figure 6.2: Random Forest Prediction Model

It is important to note that each training sample is chosen from the original dataset and is replaced back into the dataset after being chosen, so the training samples are not unique. This technique leads to different trees but with the same input size. The training sample pool was not broken into smaller pieces to train each tree with a different piece. The sample size for training each tree remains the same but the features may be different. This is called bagging. An example is shown in Figure 6.2. If the sample size for training sample 1 is n, then each tree is fed with training sets of size n containing different features. RF uses a probabilistic entropy calculation approach and hence, automatically reduces the number of features. It is considered robust due to the number of decision trees and does not suffer from the problem of overfitting since it averages individual predictions made from the set of trees which leads to minimal bias. Hence, RF have been found to be effective for solving statistical classification problems. However, it has so far not been applied to non-linear regression.

The predictor variables for RF may be numerical, categorical, continuous or discrete and output variables are either categorical or real numbers. In the case of email spam filtering, output is categorical. Suppose the input variable is vector *X* and output label is *Y,* then random forest is a classifier that learns the relationship between *X* and *Y* such that the forest (of decision trees) can make the prediction *Y* for the given input *X.* While each individual tree can be a weak learner and sensitive to noisy data, under the hypothesis that they are independent, the ensemble of trees will not be and should prove to be a strong classifier to predict the output *Y.*

The process of tree generation is the same as stated for CART; once each tree predicts an output, RF votes for each predicted result and prediction with the most votes is selected as the final prediction as shown in Figure 6.3.

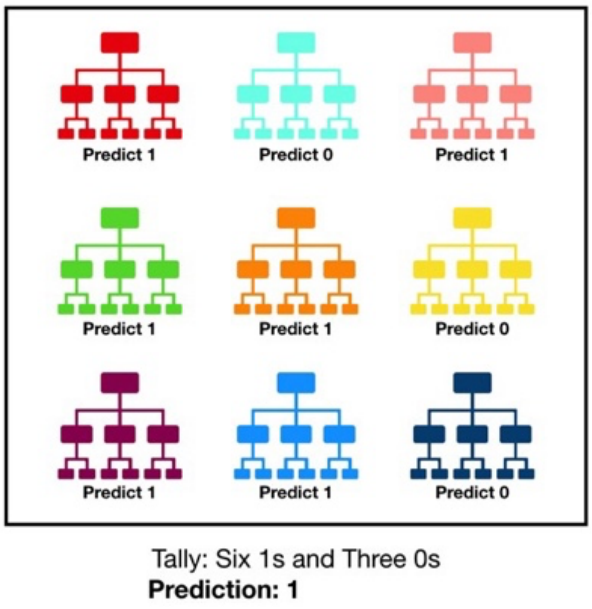


Figure 6.3: Visualisation of Random Forest prediction (Yiu, 2019)

RF has been selected for the classification of email spam due to its ease of parameterisation, robustness against overfitting, accuracy, non-sensitivity to noisy data, as well as speed and performance. It does not require pre-processing such as noise removal or scale reduction as it is not affected by scale factors due to being purely based on probability.

Akinyelu and Adewumi (2014) studied detection of phishing emails, investigating the use of Random forest machine learning algorithms to improve the accuracy of classification with a minimal number of features. RF was used on a dataset of 2000 emails from which features were extracted and utilised by the algorithm achieving high accuracy at 99.7% with very low false positives of 0.06%. Similarly, Dada and Joseph (2018b) developed a classification model based on Random Forest to increase the accuracy of prediction whole reducing the features required to achieve it. An Enron dataset of spam and ham emails of 5180 was used, prominent features were extracted and applied to an RF algorithm using a WEKA simulation environment. The result was efficient classification performance with an accuracy of 99.92% with a very low false positives rate of 0.01% and a true positive rate of 0.999%. A highly successful model was proposed by Devi (2018) who developed an attribute based random forest model as part of a framework developed for email spam detection and classification. The framework first calculated the probability of each token being spam using a Bayesian theorem, calculated the weight of each token and then the email Term Frequency Inverse Document Frequency (TFIDF) The email spam score was calculated next using genetic fitness based on which classification of emails was carried out with a random forest algorithm. On this basis, evaluation results for accuracy, weighted accuracy and the f1 score were calculated. The resultant classification shows that this proposed novel framework outperforms other existing email classification methods.

eXtreme Gradient Boosting (XGBoost)

XGBoost, a gradient boosting decision tree-based ensemble supervised machine learning algorithm developed by Chen and Guestrin (2016), was designed for classification and regression. It is a gradient boosting decision tree designed for the efficient use of computation time and memory resources and has been a recent popular choice as it has won many competitions on Kaggle and has been used in industry applications. XGBoost is a library of gradient boosting algorithms that are optimised for scalability, parallelizability, portability, quick execution and accuracy.

Boosting is a method of converting weak learners into strong learners, typically applied to decision trees and weak learners with minimum correlation between them. Boosting algorithms are Adaptive Boosting (AdaBoost), Gradient Boosting and eXtreme Gradient Boosting. AdaBoost's performance is impacted by noisy data and outliers, and the algorithm tries to overfit. Gradient boosting visualises the boosting issue as an optimisation issue by optimising the loss function based on an idea by Leo Breiman (2001). The objective is to define a loss function and minimise it. It iteratively adds each model and computes the loss which represents the residual error. This loss value is used to minimise the residual and update the final prediction label based on this learning. Suppose for the output label Y, y_i is the target label, y_i^p is the predicted label then the loss function is

$$Loss = L(y_i, y_i^p) = \sum (y_i - y_i^p)^2$$

then $\sum(y_i - y_i^p)^2$ is the sum of residual errors. The aim is to minimise the loss for the predictions by using gradient descent and updating predictions based in learning rate r, the predicted value with minimum error is

$$y_i^p = y_i^p + \mathrm{r} * \frac{\delta \sum (y_i - y_i^p)^2}{\delta y_i^p} = y_i^p - r * 2 * \sum (y_i - y_i^p)^2$$

XGBoost applies the gradient boosting framework using gradient descent architecture with enhanced algorithms and system optimisation with the base gradient boosting framework as its core. It performs system optimisation by parallelisation, which means it performs sequential tree development using parallelised implementation with multiple CPU cores for faster computing during training. XGBoost uses backward tree pruning called 'depth first approach' to improve computing time. This is necessary as the stopping criteria for tree splitting are greedy and use negative loss criteria at the point of split. The algorithm performs cache optimisation of data structures by internal buffer allocation for gradient statistics storage in order to make efficient use of hardware resources (Brownlee, 2016). It achieves algorithmic enhancements via tuning parameters such as regularisation by penalising complex models to avoid overfitting. XGBoost automatically handles missing values by learning best missing value based on training loss (‘sparsity awareness’). Sparse aware implementation of XGBoost efficient handles different types of sparse patterns in the data. It uses block structures to support tree construction using parallelisation implementation. It limits parallelisation by loop nesting by interchanging the loops for base learning where the outer loop lists the leaf node, and the inner loop determines features. A weighted quantile sketch algorithm is used for split point identification in weighted datasets and each iteration in the algorithm has a cross validation method built into it (Chen and Guestrin, 2016). It is easily integrated with GPU machines to train models with large datasets to improve performance in terms of running time. Hence, XGBoost is a tree boosting algorithm that is scalable and provides state of the art solutions for classification problems. It uses an effective combination of software and hardware methods to minimise errors and provide exceptional results in short computational time. These features make XGBoost suitable for application to email spam filtering.

Chen and Guestrin (2016) proposed XGBoost as scalable machine learning method for tree boosting. A sparsity-aware algorithm is proposed for approximate tree learning, to sparse data and build a weighted quantile sketch which enables handling weights of the instances in approximate tree learning. This system increases the speed of classification tenfold and enhances scalability memory limited settings. The results show that XGBoost is able to exploit computation and processing capabilities as well as improve speed.

Hazim, Anuar, Ab Razak, and Abdullah (2018) used Extreme Gradient Boosting (XGBoost) and a Generalized Boosting Regression method (GBM) in a supervised boosting approach on two multilingual datasets to improve the detection of spam in the mobile application marketplace. Statistics-based features were proposed and then tested on English and Malay datasets with accuracies of 87.43% and 86.13% respectively. XGBoost was found to be suitable for spam detection on the English dataset whereas GBM was more suitable for the Malay dataset. (Mussa and M. Jameel, 2019) proposed an Xtreme Gradient Boosting method for detection of SMS spam where two types of wrapper feature selection methods were used for optimal feature selection. Nine features were considered a good representation of the content of messages and the accuracy of the XGboost classifier obtained with 10-fold cross validation was 98.64%.

Dense Neural Network

This model contains Dense layers with activation as relu. The last layer has activation as sigmoid which outputs a probability between 0 and 1 with samples as spam/ham based on its confidence. To avoid overfitting, a dropout of 0.2 between layers is used. The algorithm for the DNN model is shown in Figure 6.4 below:

```
def get_dense_model(num_max):
   model = Sequential()
   model.add(Dense(2048, activation='relu', input_shape=(num_max,)))
   model.add(Dropout(0.2))
   model.add(Dense(1024, activation='relu'))
   model.add(Dropout(0.2))
   model.add(Dense(512, activation='relu'))
   model.add(Dropout(0.2))
   model.add(Dense(256, activation='relu'))
   model.add(Dropout(0.2))
   model.add(Dense(128, activation='relu'))
   model.add(Dropout(0.2))
   model.add(Dense(64, activation='relu'))
   model.add(Dropout(0.2))
   model.add(Dense(1, activation='sigmoid'))
   #model.summary()
   model.compile(loss='binary_crossentropy',
          optimizer='adam',
          metrics=['acc'])
   print('compile done')
   return model
```
Figure 6.4: Algorithm for Dense Neural Network

Convolutional Neural Network version 1 (CNNv1)

For CNN, Keras sequential model with multiple layers is used. The input into the model is the pre-processed data for which embedding vectors have been developed. It starts with an embedding layer of a vocabulary size of 1000 and embedding vector of 50 per word.
Embedding output is fed into the convolution layer with size of 64 filters and 3 channels along with activation as relu. The output of the convolution layer is fed to the GlobalMaxPooling layer to average the output of the convolution layer of 3 channels into one channel to feed it into the dense layer of size 256 and activation as relu. The output of this dense layer is fed into another dense layer of size 1 which has an activation function as sigmoid which outputs the probability of the sample email being spam or not. A dropout of size 0.2 intermediate in layers is used.

A different set of parameter values of the CNN model was used for classification to check whether it performed better or worse. It is described below.

Convolutional Neural Network version 2 (CNNv2)

Similar to CNNv1, it starts with an embedding layer of a vocabulary size of 1000; however, an embedding vector of 20 per word is used. The embedding output is fed into the convolutional layer with a size of 256 filters and 3 channels along with activation relu for this version.

- **Entropy Calculation and visualisation**

Shannon and relative entropies are calculated for each sample of ham and spam email test data. Entropy in the context of information especially which is a measure of randomness or uncertainty in data and is used in information security (L. Du, Song, and Jia, 2014).

Here two entropy-based methods, Shannon and Relative entropies are used for detection of spam emails where spammers have injected spam emails with random valid strings of text to confuse the filtering algorithms. In information theory, entropy is the amount of information required to represent the outcome of a random variable. This concept of information entropy was introduced by Claude Shannon in 1948. Shannon entropy measures the uncertainly of a random variable or distribution and is calculated using the formula:

$$H(X) = \sum_{i=1}^{n} x_i \, log\,(x_i)$$

Here, $H(X)$ is entropy of distribution X, x_i is proportion of each unique *i* in distribution X or simply probability of event *I* and n is total number of observed events or size of distribution

Relative entropy, also called as Kullback-Leibler divergence was introduced in 1951 by Solomon Kullback and Richard Leibler. It is a measure in an information theory context that compares two distributions; it is a measure of how one distribution differs from another. In this case, it is the distribution of malicious spam emails and ham emails. It is referred to as divergency for the same reason as it measures how two distributions diverge. Relative entropy for two distributions, X and Y is calculated as

$$D_{KL} = \sum_{i} x_i \; Log\left(\frac{x_i}{y_i}\right)$$

Here x_i and y_i are corresponding proportions of X and Y. One of the disadvantages of relative entropy is that it is asymmetric.

- **Classification**

Email spam classification is performed on the test data with the chosen models by machine learning algorithms. The performance of each of the models is recorded using the metrics of confusion matrix, ROC/AUC curve, and the Precision Recall curve.

- **Random String Injection**

The valid text piece is taken form William Shakespeare’s play, The tragedy of Macbeth. Random text from Macbeth of size 20% of the size of an email text is added to all datasets.
Next the steps of entropy calculation, visualisation and classification are performed. The outcome metrics are recorded.

- **Random Scrambled String Injection**

The valid text piece from Macbeth of the same size 20% of the size of email is scrambled and added to the sample test data in this part of the experiment. Entropy calculation and visualisation is performed on this test data, followed by classification of email datasets into spam and ham and performance metrics are recorded.

- **Word Cloud Generation**

Once data pre-processing of the test data was completed, word clouds were generated to identify the impact of adding valid random text to the test data. Word clouds are generated after adding random and scrambled random strings, both from Macbeth, for each dataset and then compared. The size of the words in the word cloud is indicative of the high frequency associated with those words in the dataset. Figure 6.5 shows the word clouds for the Enron test dataset for sample spam emails of size 5201. Next to each sub-figure is shown the list of words with higher frequency. The frequency of words changed with the addition of random valid text and is further changed with scrambling the added valid text.

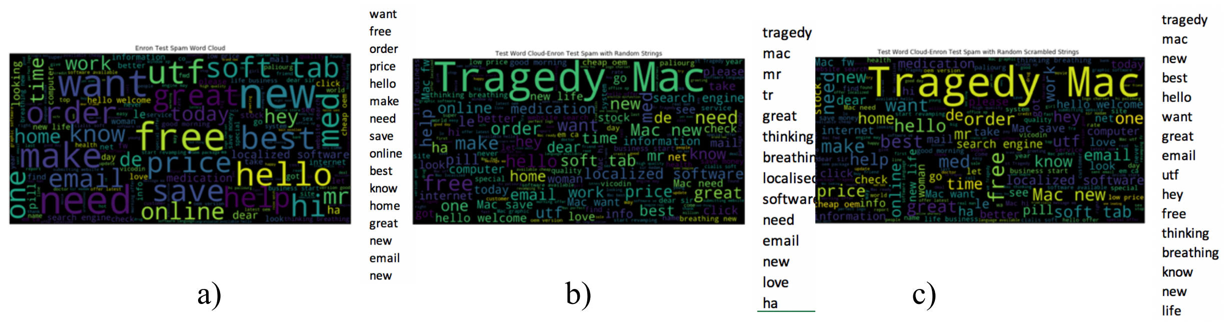


Figure 6.5: Word cloud tests for Enron dataset a) and test data b) test data with random good text c) test data with random good scrambled text.

Figure 6.6 shows the word clouds for the Ling-spam test dataset where the sample spam email size is 139. This is the smallest dataset used in this research.

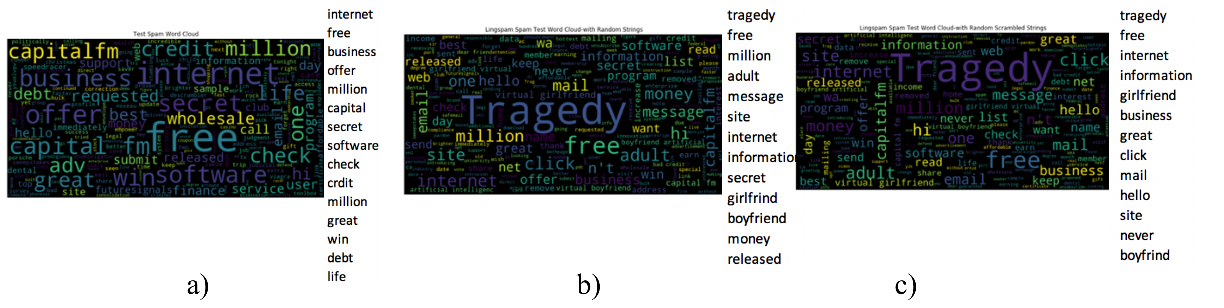


Figure 6.6: Word cloud tests for Ling-spam dataset a) test data b) test data with random good text c) test data with random good scrambled text.

In both Figures 6.5 and 6.6, the frequency of word occurrence changed significantly from Figure a) to b) and c) though the top two words with higher frequency which are common in b) and c).

Similar trends were noticed for word clouds generated for two other datasets - CSDMC2010 and TREC07.

6.3 Experimental Results and Analysis

In this section, first the experimental results for the proposed novel method for detecting email spam and ham – the entropy method - are presented. This is followed by an analysis of these results.

After feature extraction and selection was complete, for each generated model, a grid search through list of hyper parameters was performed. From the grid search, the best performing parameters for each model were recorded and the remainder discarded. For example, the following approach was taken for selecting hyper-parameters. Thus, for the model developed from the SVM, a grid search for parameter C with values [0.1, 1.0,10,100] is performed. From this grid search, it was found that 1.0 is giving superior results. Several other parameters around this value of 1.0, such as [0.5,1.0,2,5], were also tested to determine the optimal performing parameters. The results showed that 1.0 produces the best values for parameter C based on the SVM model. 3-fold cross validation was performed to remove the bias of the model, being trained by samples containing certain characteristics. The training dataset was divided into train and test sets 3 times by randomly choosing samples for each and the best performing score i recorded. The best score represents the best score of 3 folds, and weights of that results are used in the model. The best params are the ones selected after trying all parameters with 3 folds above and identifying the one which yielded the best score.

Then, probabilities for each sample of test sets were calculated used to calculate Shannon entropy and relative entropy for all selected machine learning and deep learning models. Outcomes for both entropies are shown as histogram, as distribution of entropies.
Next, random strings (Shakespeare's Macbeth text) of size 20% of the size of the email were added to each email in the test spam dataset, both types of entropies were calculated followed by visualisation as histograms. Subsequently, Shakespeare’s Macbeth text was scrambled and random strings of size 20% of the size of the email were added to each email in the test spam dataset followed by entropy calculations and visualisation.

The step by step algorithm of the approach

- ⇒ For each dataset from datasets:
- ⇒ Data Pre-processing
- ⇒ Machine learning model generation and parametrisation
- ⇒ Report Entropy (log-loss)
- ⇒ calculate probabilities for each sample (before probabilities)
- ⇒ Classification on test dataset
- ⇒ Add random strings to test datasets.
- ⇒ Classification on this modified test set using above models.

⇒ Report Entropy (log-loss)
⇒ Calculate probabilities for each test samples (After probabilities)
⇒ Calculate relative entropy using before and after probabilities
⇒ Add scrambled random strings to test sets.
⇒ Classification on this modified test set using above models.
⇒ Report Entropy (log-loss)
⇒ Calculate probabilities for each test samples (After probabilities)
⇒ Calculate relative entropy using before and after probabilities

The results of the experiments are presented as follows. For each dataset, in order starting from the smallest dataset Ling-spam, relative and Shannon entropies for each machine learning model is shown which is followed by the visualisation of entropies for ham (green) and spam (red) distributions. Next, the performance metrics showing accuracy, precision, recall and the F1 score is documented in tabular form for each model followed by ROC/AUC and PRC curves. The results are as follows.

For Ling-spam dataset, Table 6.2 shows the Shannon and relative entropies before and after random strings and after scrambled random strings, using a) machine learning models and b) deep learning models.

Table 6.2: Shannon and relative entropy for Ling-spam dataset for a) machine learning models b) deep learning models

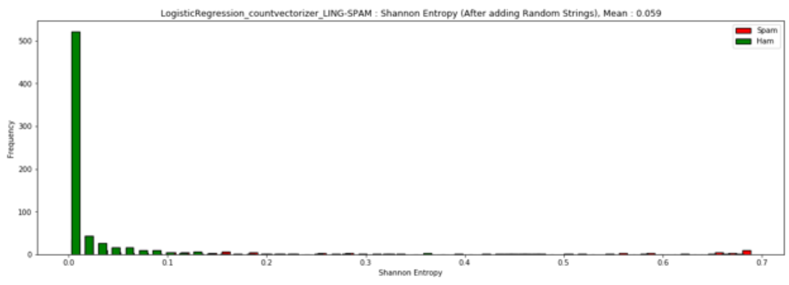
a)

	LR		SVM		XGB		NB		RF	
Entropy	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relative
BeforeString	0.06	0.116	0.062	0.101	0.185	0.166	0.025	0.094	0.17	0.1643
AfterString	0.059	0.123	0.067	0.103	0.185	0.166	0.021	0.094	0.169	0.1644
AfterScrambledString	0.167	0.578	0.008	1.013	0.431	0.455	0.448	0.454	0	1.0121

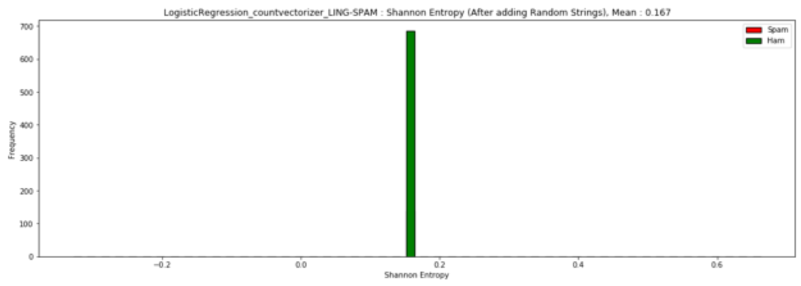
b)

	DNN		CNNV1		CNNV2	
Entropy	Shann	Relati	Shann	Relati	Shann	Relativ
BeforeString	0.035	0.156	0.037	0.206	0.037	0.184
AfterString	0.032	0.215	0.046	0.332	0.036	0.508
AfterScrambledString	0.537	1.273	0.238	2.294	0.306	2.005

Results show that Relative and Shannon entropies are not significantly different for before and after adding random strings, but both are significant after adding scrambled random strings in the test spam emails. This means that email spam detection performance of the models is significantly impacted by adding scrambled random strings.



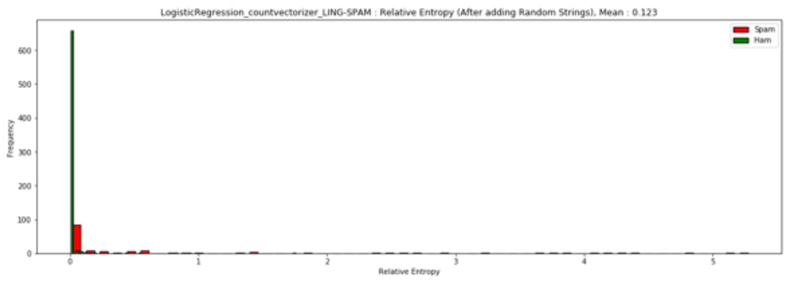
a)



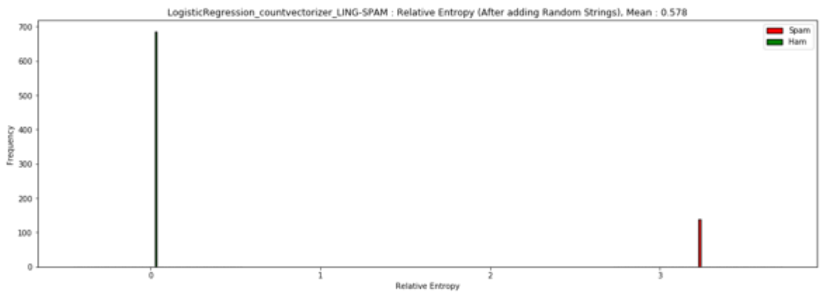
b)

Figure 6.7: Shannon entropy for Ling-spam dataset for Logistic regression machine learning model

Figure 6.7 and 6.8 show the visual representation of histograms for Shannon and relative entropies for Ling-spam a) after random strings and b) scrambled random strings.



a)



b)

Figure 6.8: Relative entropy for Ling-spam dataset for Logistic regression machine learning models

It is noticed that in case of relative entropy, spam and ham are distinct from each other in histograms but in case of Shannon entropy they overlap. Hence, it is deduced that relative entropy better separates spam and ham emails in comparison to Shannon entropy.

Table 6.3: Performance metrics for Ling-spam dataset for a) machine learning models b) deep learning models

a)

Metric->Spam	Accuracy					Precision					Recall					F1 score				
Model ->	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
Data																				
BeforeString	96	95	93	98	94	97	99	91	93	99	80	73	67	93	63	87	84	77	93	77
AfterString	96	94	93	98	94	96	100	91	95	99	79	68	67	90	63	87	81	77	93	77
AfterScrambledString	83	83	83	83	83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b)

Metric->spam	Accuracy			Precision			Recall			F1 score			ROC/AUC			PRC		
Model ->	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2
BeforeString	95.5	94	94	90	81	81	82	85	96	86	83	83	98.5	98	98	94	92	93
AfterString	94.4	91	88	96	99	100	70	49	100	81	65	47	98.5	97.6	98	94	92	91
AfterScrambledString	16.9	16.9	16.8	17	17	17	100	100	0	29	29	29	50	50	50	58	58	58

From Table 6.3, performance metrics, it becomes clear that spam detection capability of the models significantly deteriorates with scrambled random strings and not significantly with the addition of random strings. Among the machine learning models, Naïve Bayes performed best for this dataset among all the metrics used, further values of FP, FN, TP and TN in confusion matrix in Table 6.4 a) supports that NB performs the best for Ling-spam dataset.

Table 6.4: Confusion matrix for Ling-spam dataset for a) machine learning models b) deep learning models

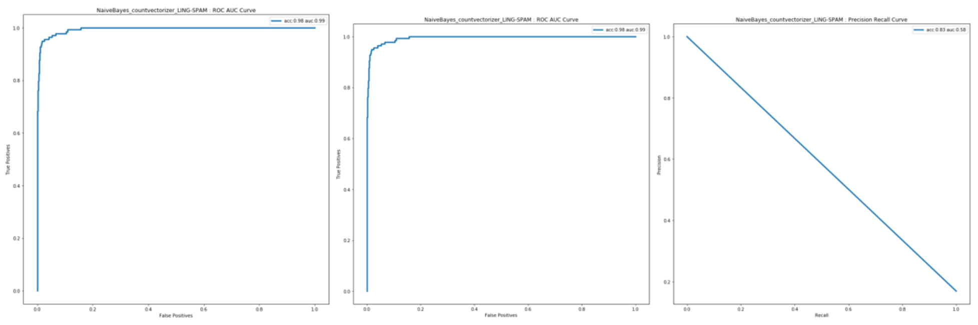
a)

	FP					FN					TP					TN				
	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
BeforeString	4	1	9	10	1	28	38	46	10	51	111	101	93	129	88	681	684	676	675	684
AfterString	4	0	9	6	1	28	44	46	14	51	110	95	93	125	88	681	685	676	679	684
AfterScrambledString	0	0	0	0	0	139	139	139	139	139	0	0	0	0	0	685	685	685	685	685

b)

	FP			FN			TP			TN		
User	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2
BeforeString	12	28	28	25	21	21	114	118	118	673	657	657
AfterString	4	1	0	42	71	96	97	68	43	681	684	684
AfterScrambledString	685	685	685	0	0	0	139	139	139	0	0	0

Among the deep learning models used, DNN outperforms both versions of CNN for all metrics as shown in Table 6.3 b). The confusion matrix for DNN in Table 6.4 b) shows significant performance for FP and TN but the values for FN and TP are slightly lower than for CNN.



a) b) c)

Figure 6.9: ROC/AUC curve a) before strings b) after strings c) after scrambled strings for Ling-spam dataset -Naive Bayes machine learning model

The ROC/AUC curves in Figures 6.9 a) and b) show that the NB models performs well at ROC with accuracy of 99% and the area under the curve at 98% for both, before and after adding random strings; however, after adding scrambled strings, performance is not equally satisfactory with accuracy of 83% and AUC at 58%.

There has been similar performance for other machine learning models as evident from Tables 6.3 and 6.4. Given the small size of the dataset, performance of deep learning models cannot be validated satisfactorily.

The next set of figures shows the performance for dataset CSDMC2010. Table 6.5 presents Shannon and Relative entropy for machine learning and deep learning models.

Table 6.5: Shannon and relative entropy for CSDMC2010 dataset for a) machine learning models b) deep learning models

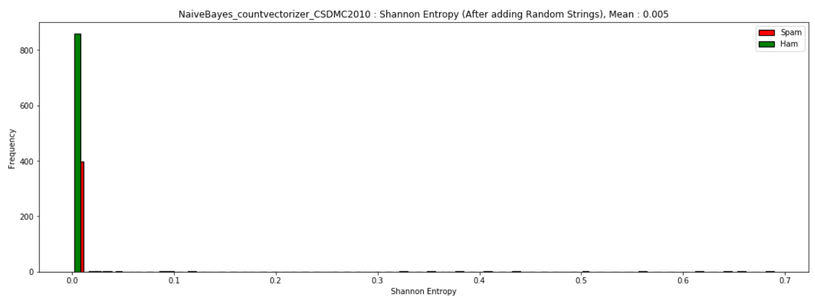
a)

	LR		SVM		XGB		NB		RF	
Entropy	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relativ
BeforeString	0.045	0.062	0.058	0.079	0.092	0.821	0.003	0.187	0.111	0.083
AfterString	0.045	0.062	0.082	0.077	0.092	0.821	0.005	0.161	0.111	0.083
AfterScrambledString	0.679	0.767	0.657	0.826	0.237	1.887	0.624	0.629	0.638	0.872

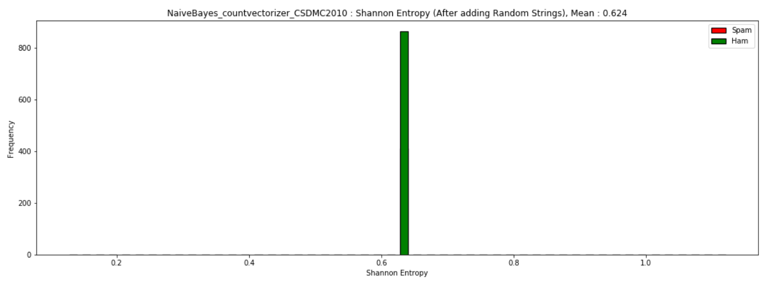
b)

	DNN		CNNV1		CNNV2	
	Shann	Relati	Shann	Relati	Shann	Relativ
BeforeString	0.031	0.097	0.069	0.074	0.056	0.078
AfterString	0.037	0.1	0.079	0.076	0.067	0.08
AfterScrambledString	0.509	1.282	0.657	0.852	0.652	0.853

The entropy values show a similar trend to the Ling-spam dataset, where both Shannon and Relative entropy values are similar before and after adding strings but are higher after adding random scrambled strings for all models.



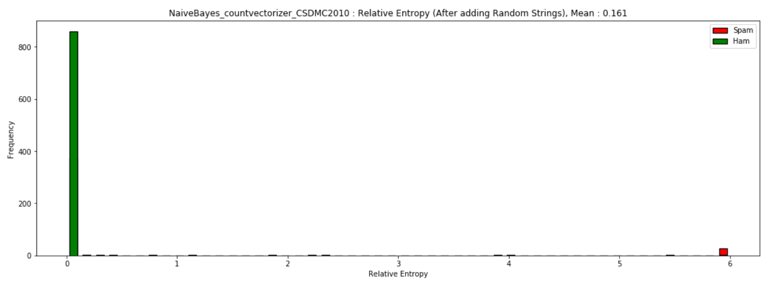
a)



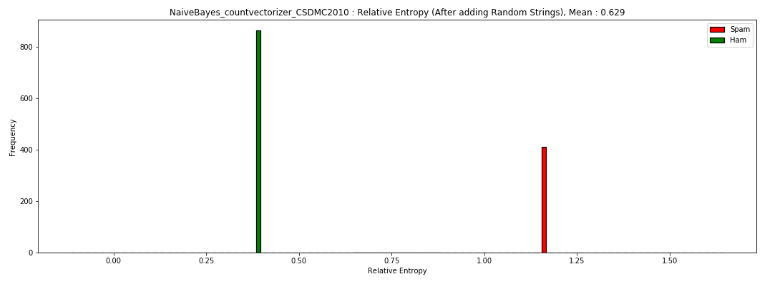
b)

Figure 6.10: Shannon entropy for CSDMC2010 dataset for Naïve Bayes machine learning model a) after adding strings b) after adding scrambled strings

From Figure 6.10, Shannon entropy after adding random strings shows that entropy value for spam is lower than ham; however, they overlap, hence are not well suited for classification. The Shannon entropy value for ham after adding scrambled strings is significantly higher than spam which is almost 0.



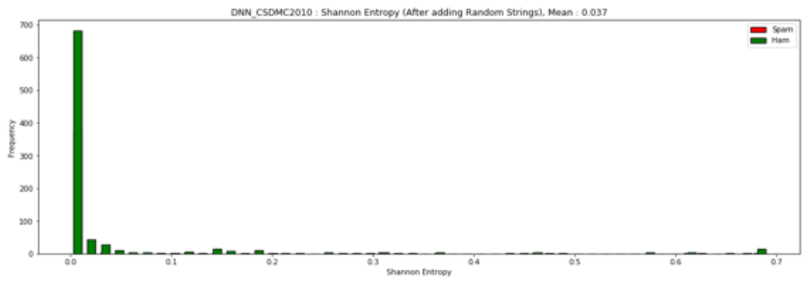
a)



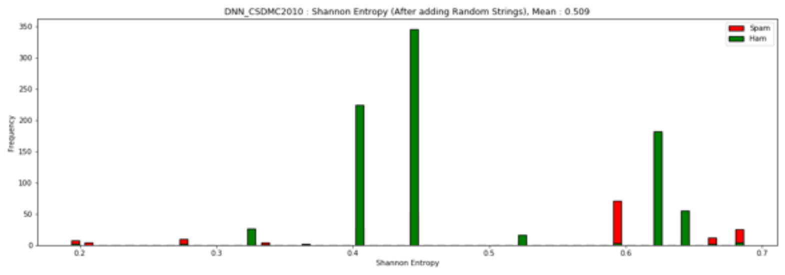
b)

Figure 6.11: Relative entropy for CSDMC2010 dataset for Naïve Bayes machine learning model a) after adding strings b) after adding scrambled strings

Relative entropy values for ham are higher than spam values for both, after strings and after scrambled strings are added and for spam are distinctly lower for both. There is a clear distinction between spam and ham in both cases for relative entropy values as shown in histograms in Figures 6.11 a) and b) for the NB model for the CSDMC dataset.
Using deep learning models, the Shannon entropy values for ham are higher than spam values for both after strings and after scrambled strings; however, spam is not efficiently separated from ham as shown in Figure 6.12.



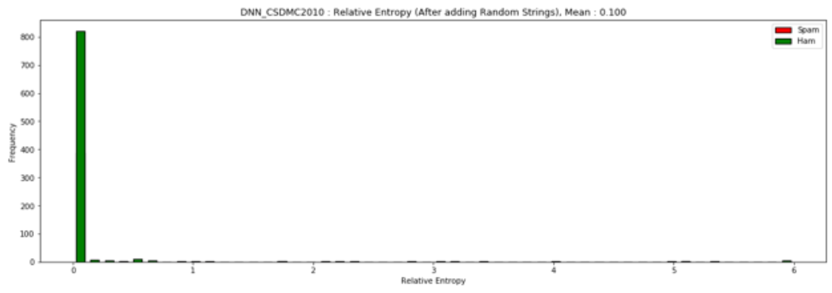
a)



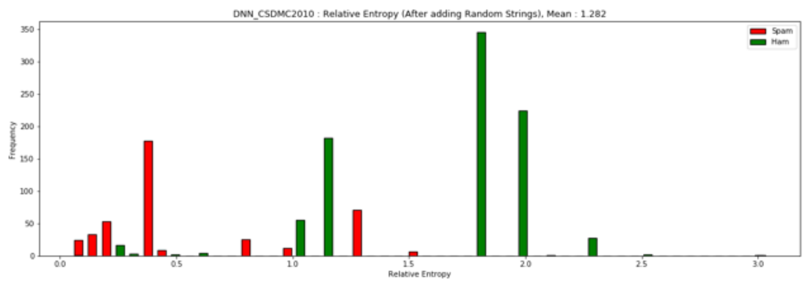
b)

Figure 6.12: Shannon entropy for CSDMC2010 dataset for DNN deep learning model a) after adding strings b) after adding scrambled strings

Shannon entropy values for ham and spam using the deep learning model (DNN) shows better distinction than machine learning models, evident from Figures 6.10 and 6.12.



a)



b)

Figure 6.13: Relative entropy for CSDMC2010 dataset for DNN deep learning model a) after adding strings b) after adding scrambled strings

Relative entropy values for CSDMS2010 dataset using deep learning model (DNN) in Figure 6.13 show a similar trend to the NB machine learning model in Figure 6.11; however, the distinction between ham and spam is clearer using DNN.

Table 6.6: Performance metrics for CSDMC2010 dataset for a) machine learning models b) deep learning models

a)

Metric->Spam	Accuracy					Precision					Recall					F1 score				
Model ->	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
Data																				
BeforeString	98	98	97	96	97	99	99	96	99	97	96	96	95	90	96	97	97	96	94	97
AfterString	98	98	97	96	97	99	99	96	99	97	96	96	95	91	96	97	97	96	95	97
AfterScrambledString	32	32	32	67	32	32	32	32	0	32	100	100	100	0	100	49	49	49	0	49

b)

Metric->spam	Accuracy			Precision			Recall			F1 score		
Model ->	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2
BeforeString	97.5	97	96.7	98	96	995	94	95	94	96	95	95
AfterString	97.5	97	99.5	97	96	95	95	95	95	96	95	95
AfterScrambledString	25.3	31.5	32.3	26	31	32	72	92	100	38	47	49

Spam detection performance for all machine learning models is closely comparable except NB which is not performing well for this dataset as shown in Table 6.6 a). For deep learning models, CNNv2 outperforms the other two models though it is close to CNNv1 shown in Table 6.6 b). However, the confusion matrix reflects high false positives for CNNv2 after adding scrambled random strings as in Table 6.7 b).

Table 6.7: Confusion matrix for CSDMC2010 dataset for a) machine learning models b) deep learning models

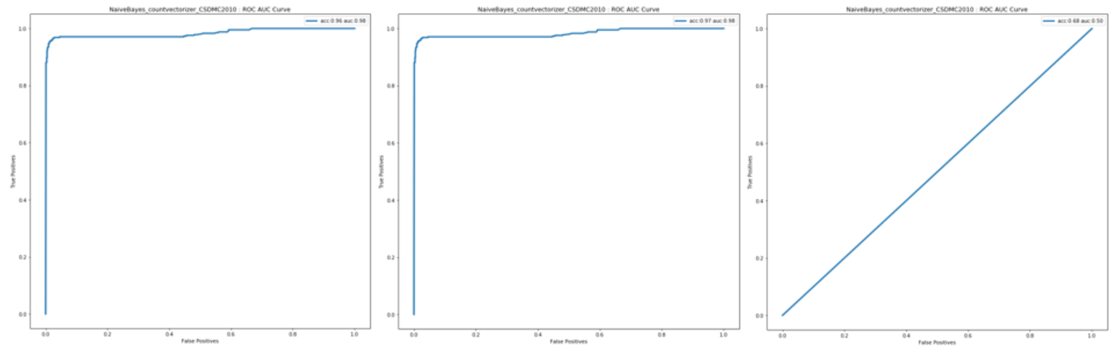
a)

	FP					FN					TP					TN				
	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
BeforeString	6	5	15	4	12	16	18	20	41	15	396	384	392	371	397	857	858	848	859	851
AfterString	6	6	15	4	12	16	17	20	36	15	396	385	392	376	397	857	857	848	859	851
AfterScrambledString	6	5	15	0	11	22	18	0	40	0	390	412	412	372	412	857	858	848	853	852

b)

	FP			FN			TP			TN		
User	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2
BeforeString	8	16	19	24	22	23	388	390	389	855	847	844
AfterString	12	16	20	20	22	21	392	390	391	851	847	843
AfterScrambledString	837	842	862	115	32	1	297	390	411	26	21	1

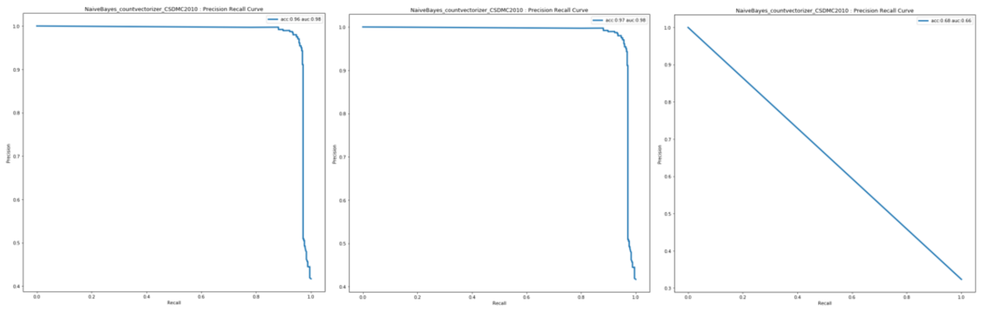
Although the NB did not return high performance for some metrics, the ROC/AUC curve shows accuracy of 96% and AUC of 98% f before and after the addition of random strings as shown in Figure 6.14 a) and b). However, the performance is significantly impacted after adding scrambled random strings.



a) b) c)

Figure 6.14: ROC/AUC curve CSDMC2010 dataset for a) before strings b) after strings c) after scrambled strings NB machine learning model

Figure 6.15 - PRC curve performance - shows trends similar to the ROC/AUC curve with 96% accuracy and AUC 98% for before strings, 97% accuracy and AUC 98% for after adding random strings and only 68% accuracy and AUC 66% for after adding random scrambled strings.



a) b) c)

Figure 6.15: PRC curve CSDMC2010 dataset for a) before strings b) after strings c) after scrambled strings NB machine learning model

For the Enron dataset, relative entropy shows an increasing trend in values, being lowest before strings are added, slightly higher after adding random strings and highest after adding scrambled random strings for all models as presented in Table 6.8.

Table 6.8: Shannon and relative entropy for Enron dataset for a) machine learning models b) deep learning models

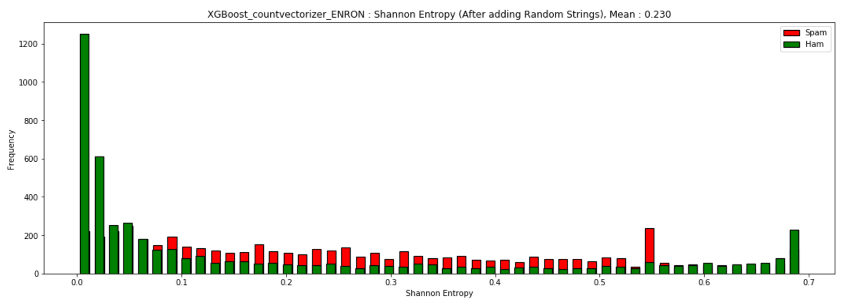
a)

	LR		SVM		XGB		NB		RF	
	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relati
BeforeString	0.06	0.055	0.052	0.055	0.23	0.139	0.012	0.06	0.248	0.132
AfterString	0.061	0.054	0.088	0.063	0.23	0.139	0.012	0.061	0.258	0.135
AfterScrambledString	0.034	1.146	0.028	2.635	0.543	0.844	0.693	0.693	0	2.916

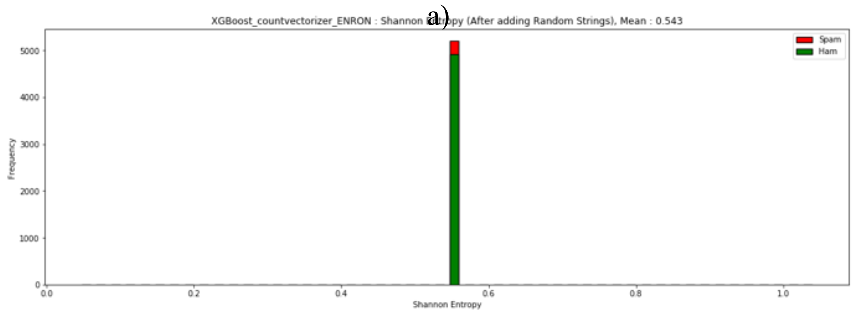
b)

Enron	DNN		CNNV1		CNNV2	
	Shanno	Relative	hanno	Relative	hanno	Relative
BeforeString	0.002	0.378	0.042	0.712	0.624	0.852
AfterString	8E-04	0.378	0.04	0.734	0.624	0.852
AfterScrambledString	0.68	0.711	0.523	0.868	0.679	0.703

The best performance is by the SVM for the machine leaning models and CNNv1 for the deep learning models. Figure 6.16 shows the histograms for Shannon entropy values for ham and spam emails.



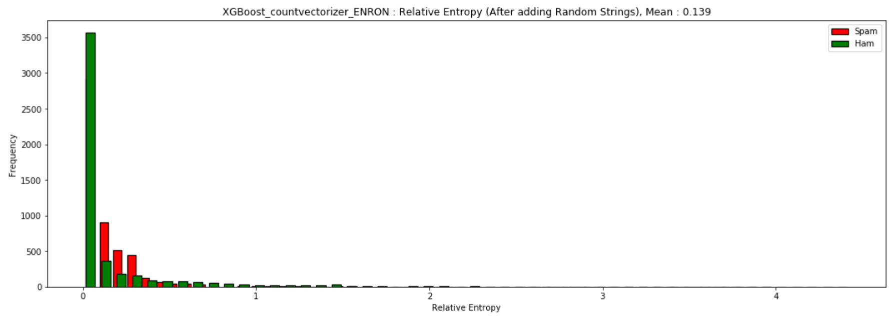
a)



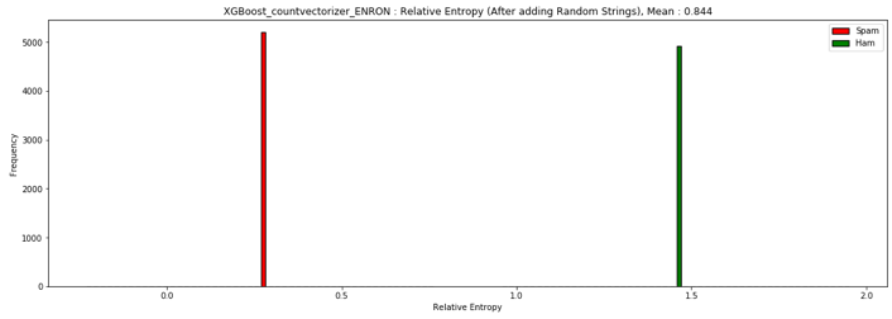
b)

Figure 6.16: Shannon entropy for Enron dataset for XGB machine learning model a) after adding strings b) after adding scrambled strings

For the Enron dataset, the Shannon entropy values in Figure 6.16 do not contribute significantly to separating the ham and spam emails in both cases of a) after adding random strings and b) after adding scrambled random strings to the test data samples.



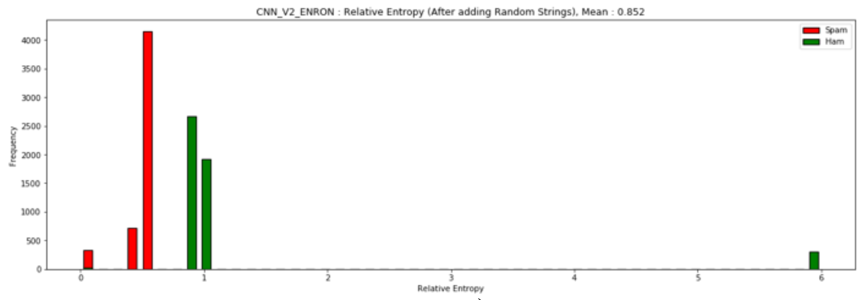
a)



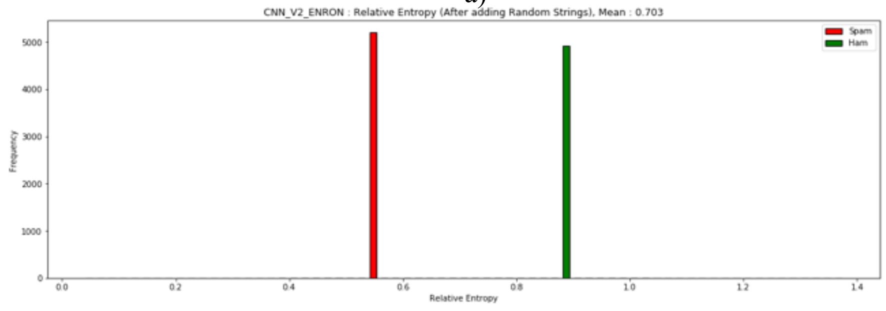
b)

Figure 6.17: Relative entropy for Enron dataset for XGB machine learning model a) after adding strings b) after adding scrambled strings

Figure 6.17 a) shows an overlap of ham and spam values for relative entropies whereas Figure 6.17 b) values show clear distinctions between spam and ham emails for the XGBoost machine learning model. This trend is similar for other models.



a)



b)

Figure 6.18: Relative entropy for Enron dataset for CNNv2 deep learning model a) after adding strings b) after adding scrambled strings

Contrary to machine learning models performance, deep learning model performance in Figure 6.18 highlights that relative entropy values separate ham and spam emails efficiently for a) after adding random valid strings and b) after adding scrambled random valid strings.

Table 6.9: Performance metrics for Enron dataset for a) machine learning models b) deep learning models

a)

Metric->	Accuracy					Precision					Recall					F1 score				
Model ->	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
Data																				
BeforeString	98	98	96	98	97	97	98	94	99	97	99	99	98	98	97	98	98	96	98	97
AfterString	98	98	96	98	97	98	98	94	99	97	99	98	98	98	97	98	98	96	98	97
AfterScrambledString	51	51	51	51	51	51	51	51	51	51	100	100	100	100	100	68	68	68	68	68

b)

Metric->forSpam	Accuracy			Precision			Recall			F1 score		
Model ->	DNN	CNNv	CNNv	DNN	CNNv	CNNv	DNN	CNNv	CNNv	DNN	CNNv	CNNv
BeforeString	93.5	85	52	90	83	51	98	91	100	94	87	68
AfterString	93.5	85	52	90	83	52	98	91	100	94	87	68
AfterScrambledString	0.49	51	51	0	51	51	0	100	100	0	68	68

For the Enron dataset, all machine learning models show comparable performance for classification metrics (Table 6.9 a) though the SVM outperformed others in relative entropy. For deep learning models, the CNN outperformed the DNN as shown in Figure 6.9 b) with added scrambled random strings; however, overall the DNN performed better than CNN versions.

Table 6.10: Confusion matrix for Enron dataset for a) machine learning models b) deep learning models

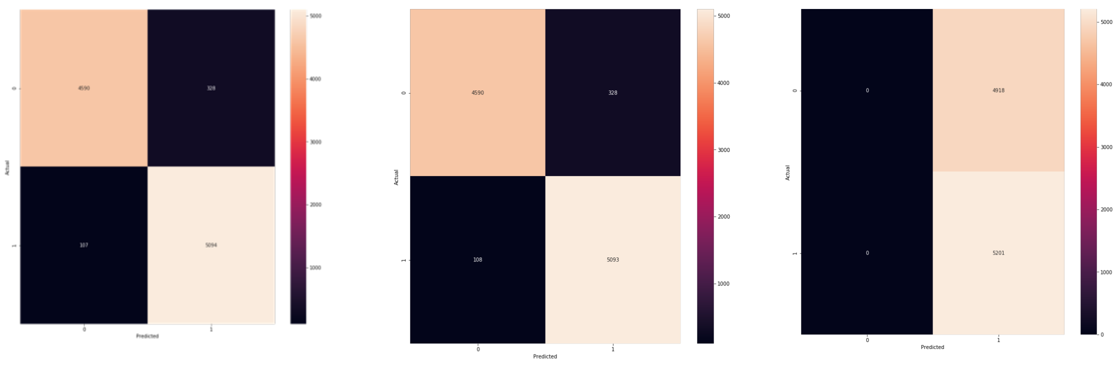
a)

	FP					FN					TP					TN				
	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
BeforeString	133	122	328	64	162	63	73	107	108	161	5183	5128	5094	5093	5040	4785	4796	4590	4854	4756
AfterString	109	100	328	63	156	76	103	108	109	162	5125	5098	5093	5092	5039	4809	4918	4590	4854	4762
AfterScrambledString	4918	4918	4918	4918	4918	0	0	0	0	0	5201	5201	5201	5201	5201	0	0	0	0	0

b)

	FP			FN			TP			TN		
	DNN	CNNv	CNNv	DNN	CNNv	CNNv	DNN	CNNv	CNNv	DNN	CNNv	CNNv2
BeforeString	560	969	4893	91	458	9	5110	4743	5192	4385	3949	25
AfterString	541	963	4889	109	480	9	5092	4721	5192	4377	3935	29
AfterScrambledString	0	4918	4918	5201	0	0	0	5201	5201	4918	0	0

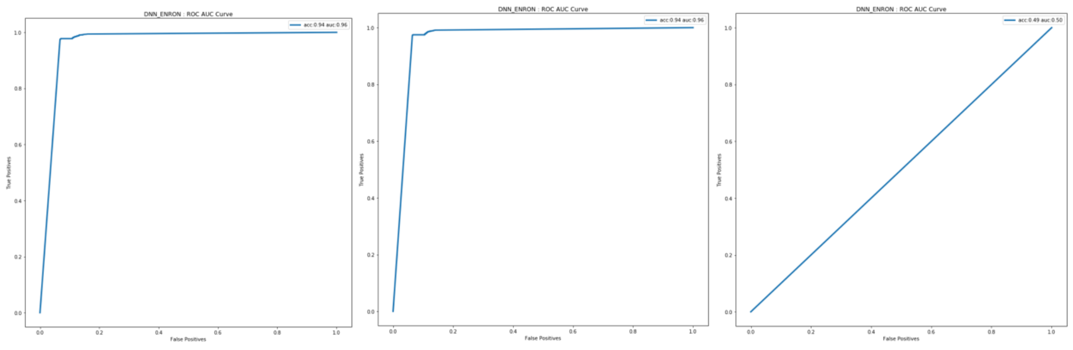
Table 6.10 reports that adding scrambled random strings significantly increases the false positives in turn reducing the true negative values for all machine learning models and for CNN deep learning models. This means that adding such strings impact the performance of the models.



a) b) c)

Figure 6.19: Confusion matrix for Enron dataset for XGB machine learning model for a) before strings b) after strings c) after scrambled strings

Figure 6.19 is a visual representation of the confusion matrix for the XGboost machine learning model where c) shows the high false positives leading to zero true negatives. The ROC/AUC curve for the DNN shown in Figure 6.20 support the performance noted from other metrics for all cases.



a) b) c)

Figure 6.20: ROC/AUC curve for Enron dataset for DNN deep learning model a) before strings b) after strings c) after scrambled strings

The PRC curve (Figure 6.21) for the DNN for Enron shows an accuracy of 94% and an AUC of 97% before and after strings but just 49% accuracy after adding random strings, with an AUC of 76%.

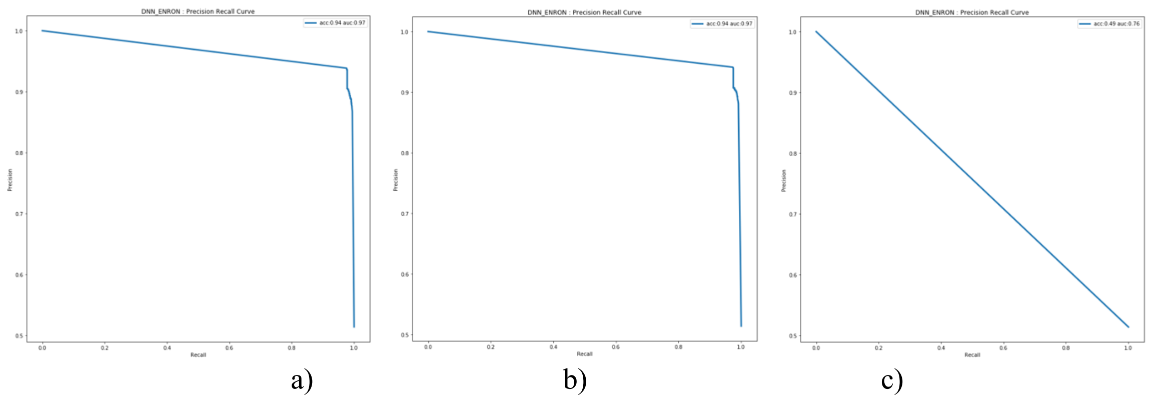


Figure 6.21: PRC curve for Enron dataset for DNN deep learning model a) before strings b) after strings c) after scrambled strings

The following set of figures show the results of the TREC07 dataset which is the largest dataset containing >75,000 emails with test data containing 10040 spam and 5044 ham emails.

Table 6.11: Shannon and relative entropy for TREC07 dataset for a) machine learning models b) deep learning models

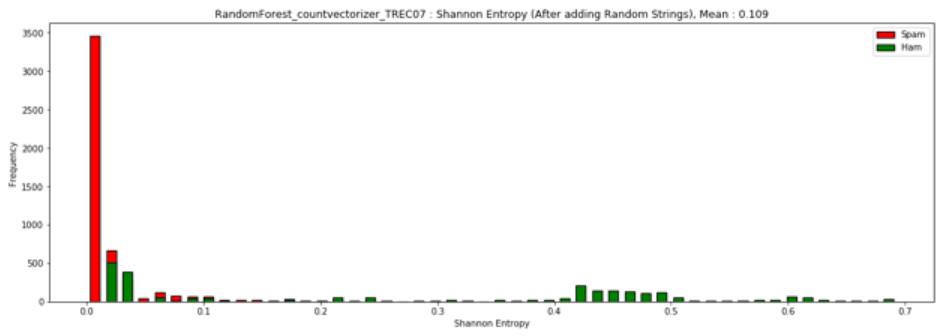
a)

	LR		SVM		XGB		NB		RF	
entropy	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relati	Shann	Relativ
BeforeString	0.01	0.016	0.011	0.013	0.018	0.031	0.007	0.037	0.103	0.045
AfterString	0.01	0.016	0.018	0.015	0.018	0.031	0.004	0.035	0.109	0.048
AfterScrambledString	0.463	0.712	0.671	0.645	0.558	1.027	0.637	0.637	0.688	0.731

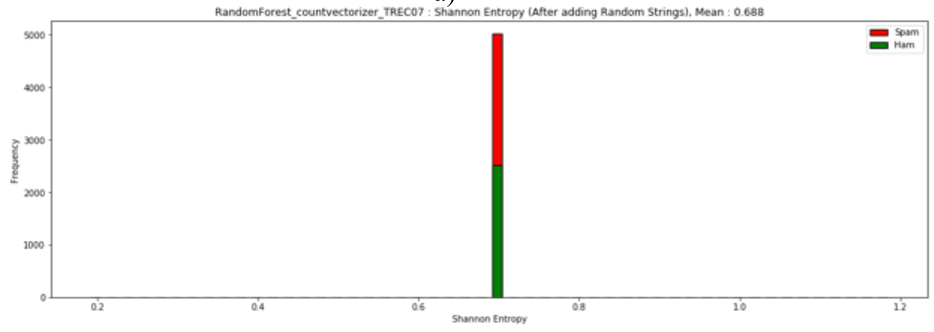
b)

TREC07	DNN		CNNV1		CNNV2	
	Shanno	Relativ	hanno	Relativ	hanno	Relative
BeforeString	0.002	0.03	0.005	0.921	0	0.079
AfterString	0.001	0.034	0.005	0.922	0	0.076
AfterScrambledString	0.044	1.978	0.005	0.922	0.041	2.534

Entropy values for the TREC07 dataset show an interesting change where Shannon entropy values are similar to Relative entropies as shown in Table 6.11. Similar to random entropies, the Shannon entropy values are lowest before adding strings, slightly higher after adding random strings and highest once scrambled random strings have been added. The values of random entropies for the data with scrambled random strings added are significantly higher which is same as the pattern for other datasets.



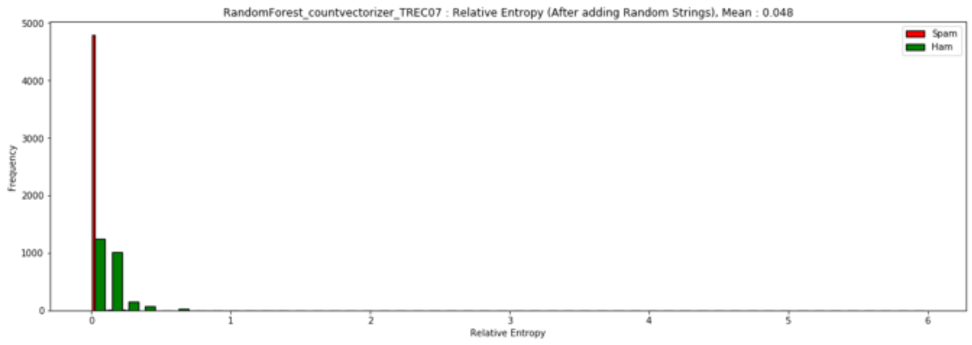
a)



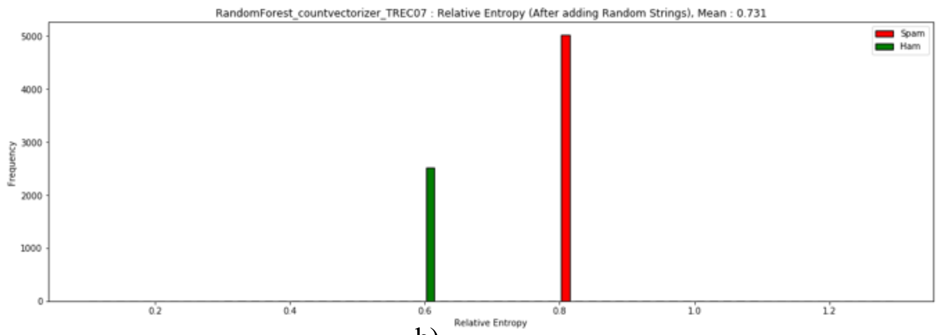
b)

Figure 6.22: Shannon entropy for TREC07 dataset for SVM machine learning model a) after adding strings b) after adding scrambled strings

Shannon entropy values for ham and spam are again overlapping in Figure 6.22 for both a) after adding random strings and b) after adding scrambled random strings and they do not effectively separate these for the TREC07 dataset.



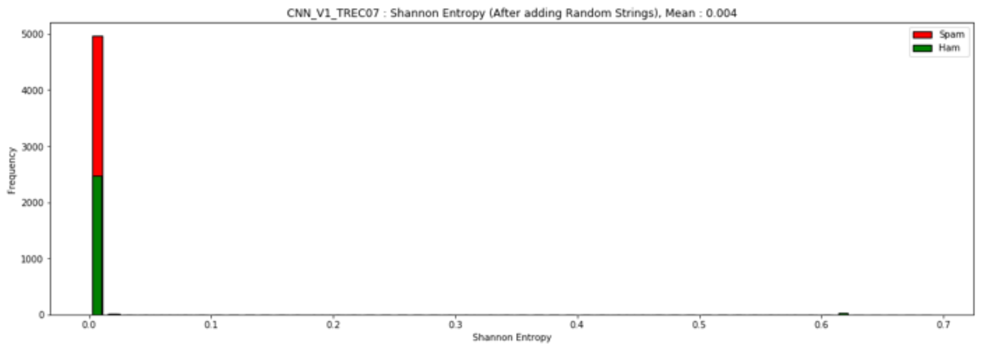
a)



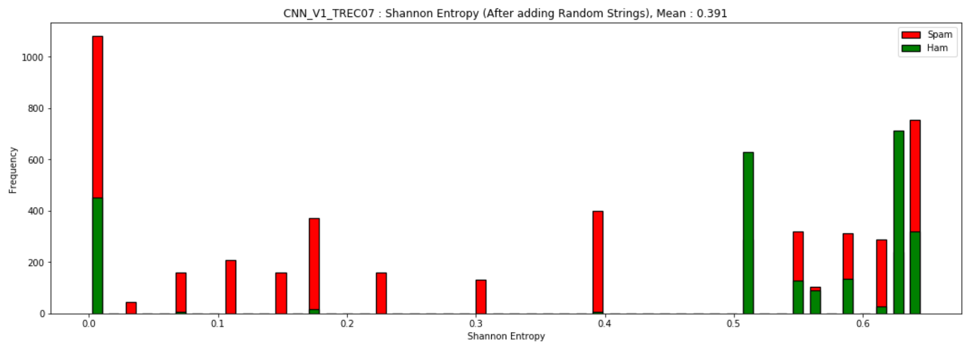
b)

Figure 6.23: Relative entropy for TREC07 dataset for SVM machine learning model a) after adding strings b) after adding scrambled strings

Figure 6.23 a) demonstrates that the SVM machine learning model returns results where the distributions are placed close to each other after adding random strings to the dataset, although relative entropies for spam emails are higher than for ham emails. When adding scrambled random strings to the dataset, however, (Figure 6.23 b)) relative entropy values show clear separation between the two distributions with low entropy values for ham and high ones for spam.



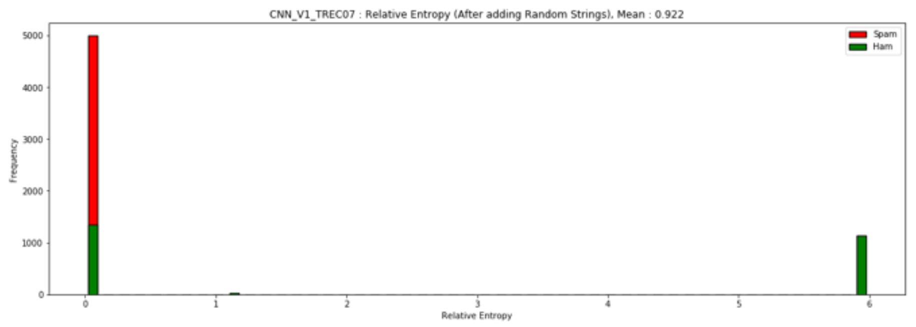
a)



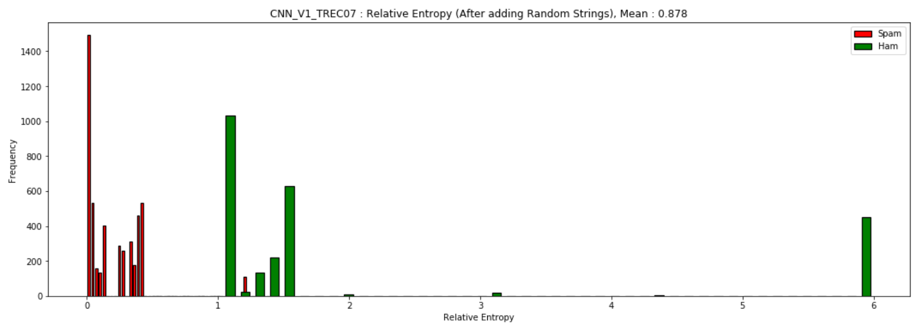
b)

Figure 6.24: Shannon entropy for TREC07 dataset for CNNv1 deep learning model a) after adding strings b) after adding scrambled strings

Using the DNN, Shannon entropy values for the distributions of ham and spam for the TREC07 dataset has higher overlap than machine learning methods for both scenarios – adding random strings to the dataset as in Figure 6.24 a) and adding scrambled random strings (Figure 6.24 b).



a)



b)

Figure 6.25: Relative entropy for TREC07 dataset for CNNv1 deep learning model a) after adding strings b) after adding scrambled strings

Relative entropies for ham and spam distributions still show overlap for adding random strings to the dataset as in Figure 6.25 a) though some ham emails are far apart from spam distribution. For the case of adding scrambled random strings as shown in Figure 6.25 b), the relative entropy separates the two distributions with little overlap where ham emails have similar entropy values to spam emails.

Table 6.12: Performance metrics for TREC07 dataset for a) machine learning models b) deep learning models

a)

Metric->	Accuracy					Precision					Recall					F1 score				
Model ->	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
Data																				
BeforeString	99.4	99.4	99	99.9	99.9	99	99	99	99	100	100	100	100	99	100	100	100	99	99	100
AfterString	99	99	99	99	99.7	99	99	98	99	100	100	100	100	99	100	100	100	99	99	100
AfterScrambledString	66	67	33	67	33.5	67	67	0	67	0	100	100	0	100	0	80	80	0	80	0

b)

Metric->	Accuracy			Precision			Recall			F1 score		
Model ->	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2
BeforeString	99	84	98.6	99	81	99	99	100	99	100	89	99
AfterString	99	84	98.7	99	81	99	99	100	99	100	89	99
AfterScrambledString	63	63	50	66	65	60	91	95	73	76	77	66

For the TREC07 dataset, machine leaning as well as deep learning models have a comparable performance. LR, SVM and NB among machine learning models perform well with 99% rates for before and after adding random strings; however, the performance drops by about 30% by adding scrambled strings. This is similar to the DNN and both versions of CNN with 25% lower performance after adding scrambled strings to the TREC07 dataset as shown in Table 6.12.

Table 6.13: Confusion matrix for TREC07 dataset for a) machine learning models b) deep learning models

a)

	FP					FN					TP					TN				
	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF	LR	SVM	XGB	NB	RF
BeforeString	37	34	79	36	42	8	8	35	67	7	5012	5012	4985	4953	7	2485	2488	2443	2486	2480
AfterString	40	36	145	40	73	6	7	25	27	6	5014	5012	4995	4993	6	2482	2486	2377	2482	2449
AfterScrambledString	2522	2522	2522	2522	2522	0	0	0	0	0	5020	5020	5020	5020	0	0	0	0	0	0

b)

	FP			FN			TP			TN		
	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2	DNN	CNNv1	CNNv2
BeforeString	43	1171	41	8	16	58	5012	5004	4962	2479	1351	2481
AfterString	44	1173	41	4	15	55	5016	5005	4965	2478	1349	2481
AfterScrambledString	2392	2521	2418	429	273	1369	4591	4747	3651	130	1	104

The confusion matrices for all models show that the performance declines with adding scrambled strings to the TREC07 dataset as there are high numbers of false positives shown in Table 6.13.

Among the deep learning models, CNNv1 performed least well; Figure 6.26 shows its ROC/AUC curve for a) before valid strings b) after adding random strings and c) after adding scrambled strings. We chose to show the curves for this model as the DNN and CNNv2 curves are similar to Figure 6.29 a) and b).

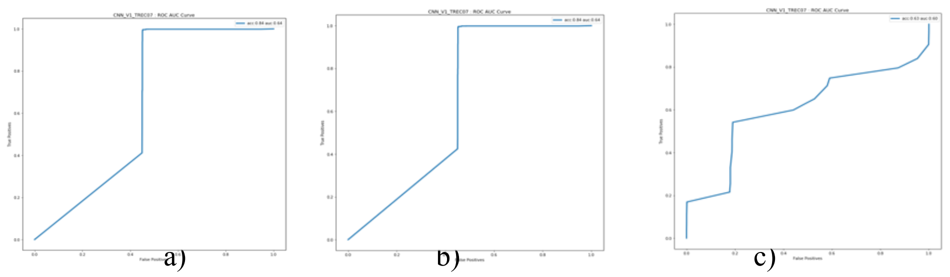


Figure 6.26: ROC/AUC curve for TREC07 dataset for CNNv1 deep learning model a) before valid strings b) after adding random strings c) after adding scrambled strings

The PRC curve for machine learning models is similar to Figure 6.15; Figure 6.27 shows the PRC curve for CNNv1 with an AUC of 64% for a) before valid strings and b) after adding random strings and AUC of 60% c) after adding scrambled strings.

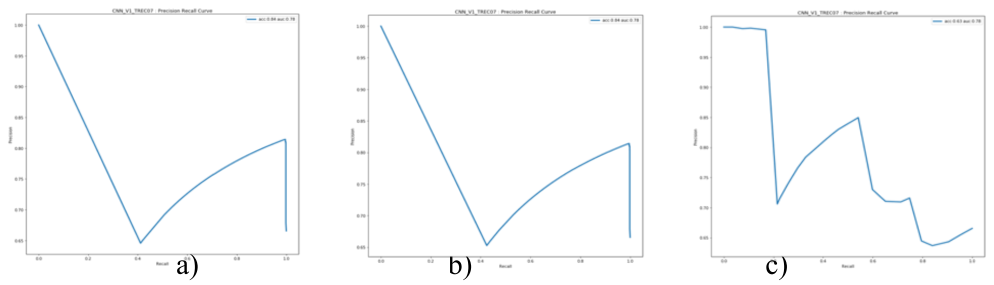


Figure 6.27: PRC curve for TREC07 dataset for CNNv1 deep learning model a) before valid strings b) after adding random strings c) after adding scrambled strings

From the results presented above, it can be concluded that spammers can successfully deceive email spam classification filters by injecting random scrambled valid strings into email spam. Relative entropy distributions using machine and deep learning models can be used to detect such injections though here deep learning models are outperformed by machine learning models unless the dataset size is large. This is in line with the characteristic for deep learning techniques as they require large dataset to produce quality performance.

6.4 Topic-based Classification at Client Side

Feature selection is one of the most important steps of email spam filtering where the content of the email message is examined to identify the features that make substantial contributions to making a decision whether a certain email is ham or spam. An email message typically consists of header and body. The majority of the literature has focused on the body of the email message to identify such features. The header field has also gathered some attention in detecting email spam (Isacenkova & Balzarotti, 2014; Khamis et al., 2020; Z. Yang et al., 2006). The rationale for filtering email spam using the header is to reduce computational complexity. The basic attributes of the header field are the sender's name, sender's email address and IP (if available), sending date, return path, received, x-mailer, number of receivers (in some cases) and subject. Sometimes, it is easy to pick out email spam by merely checking a few keywords in the subject of the email rather than scanning the header or the full content of the email body. However, there is no evidence from literature that the subject field has been used as a feature in email spam filtering. (Takesue, 2010) has considered the subject header of an email message as part of a cascading filter where body and header fields are also considered.

In this section, an email spam filter is proposed that considers features in the subject header of an email message. Once the feature extraction is done, selection is performed using GloVe embeddings. Model generation is performed using a series of machine and deep learning models explained in Chapter 4/5 and Section 6.2. The evaluation of the model and classification od test data is done using all four datasets explained in Section 6.1.

Methodology:

The steps listed and explained below are performed. Figure 6.28 shows the overall architecture for topic-based classification

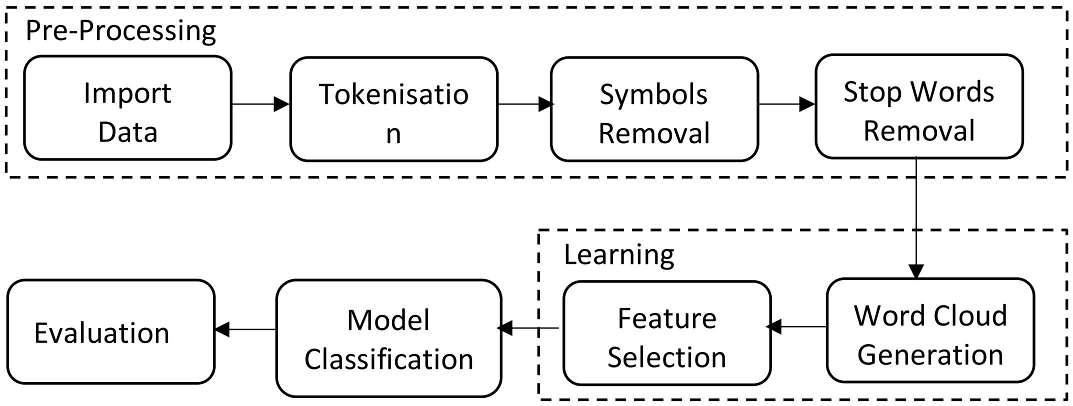


Figure 6.28: Overall architecture of topic-based classification

Pre-processing

The words from the subject field are parsed and cleaned using tokenisation and lemmatisation, brought to their root forms, punctuation marks, symbols and stop words are removed. The word 'subject' is removed as it is a frequently occurring word and would impact the total spam probability especially if the number of words in each email feature set is small.

Generate word clouds

Word clouds are generated for training samples and test samples. Further word clouds are generated for the ham and spam for each of the training and test samples to visualise the distinction of words in each of them.

Feature selection

Feature extraction and selection is performed with global vectors for word representation (GloVe) and for word embeddings which is a weighted least squares model which performs training based on global word to word co-occurrence and builds a meaningful substructure of word vector spaces. Here, this method is using a glove dataset with a dimension of 300 resulting in 40,000-word vectors. GloVe embeddings from (Pennington, Socher, & Manning, 2014) are used by first loading glove files from the GloVe Dataset and then maintaining an embeddings dictionary that maps from words to embedding vectors of length 300. Next, the email text is converted to embedding vectors. To do so, for each word in the email, an embedding from the embedding dictionary is found and then vectors of each word are normalised to create one vector which represents the contents of that mail text.

Model generation

Classification models are generated using the chosen machine and deep learning algorithms. For each model, hyper parameters are selected by performing a grid search to find the optimum performing parameters.

Classification

Classification of the training samples is performed using the selected parameters for all of the models and performance of each of the model is recorded.

Evaluation

The generated models are evaluated on the test samples for each of the four datasets and the performance metrics recorded and presented in the next section.

For each dataset:

⇒ Pre-processing to Clean Data
⇒ Create word clouds
⇒ Transform data using word embeddings
⇒ Try various machine learning models namely SVC, Random Forest, Gradient Boosting, XGBoost and deep learning models namely DNN and CNN.
⇒ Classification on train data
⇒ Evaluation and Classification on test data
⇒ Record Performance metrics

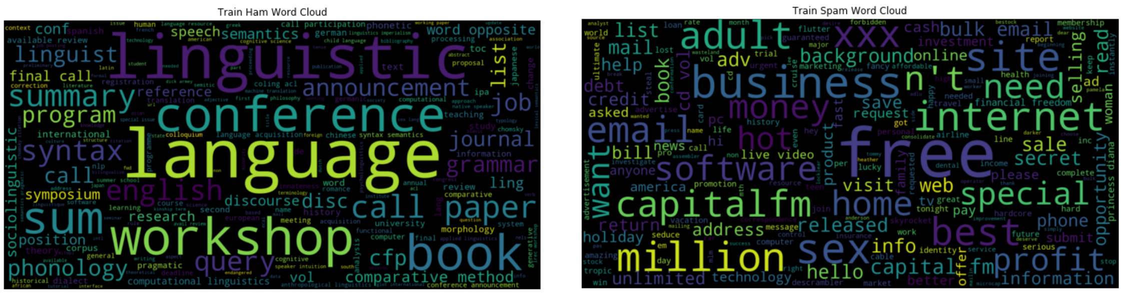
6.5 Experiment Results and Analysis on Topic-based Classification

In this section the experimental results are presented. Which are followed by an analysis of the results. It was found that the GloVe models outperforms other baseline models that often have smaller vector sizes and smaller datasets. For each dataset, GloVe embeddings of dimension 300 provided by (Pennington et al., 2014) were loaded, although dimensions of 50,100 and 200 are also available. After training several models, it was found that GloVe provides superior embeddings for words. Embedding of size 300 means that the Glove file has a vector of size 300 for the most common words. Each vector has floats of a length of 300. The total number of words found in the dictionary of the datasets is shown in Table 6.14.

Table 6.14: Total number of Words in dictionary found from the Subject Field

TREC07	19661
Enron	12248
CSDMS2010	3721
LingSpam	2332

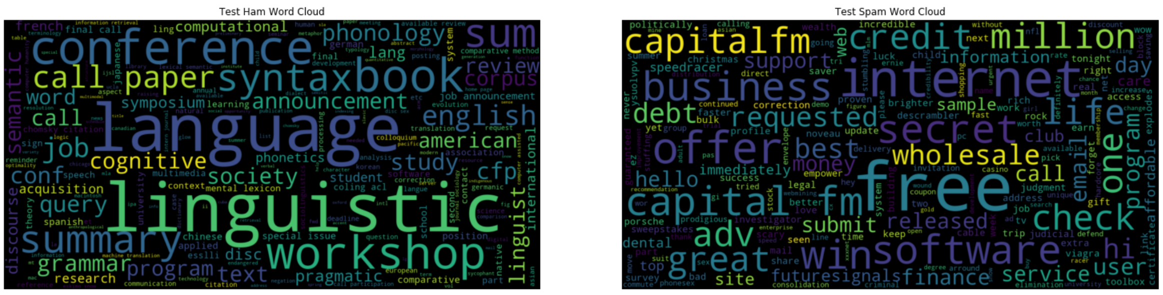
Next, word clouds were generated for each dataset. Figure 6.29 shows the word clouds for training samples for a) ham and b) spam. It is noted that the words in ham cloud (Figure 6.29 a) show ‘conference’, language, workshop, ‘linguistic’ and similarly acceptable words whereas Figure 6.29 b) shows the words such as ‘free’, ‘million’, ‘adult’, ‘business’, ‘sex’, ‘internet’ etc.



a) b)

Figure 6.29: Ling-spam Subject Field Word Clouds for Train Data

Figure 6.30 show the word clouds for test data of Ling-spam dataset with similar trends of ham and spam words in Figure a) and b) respectively.

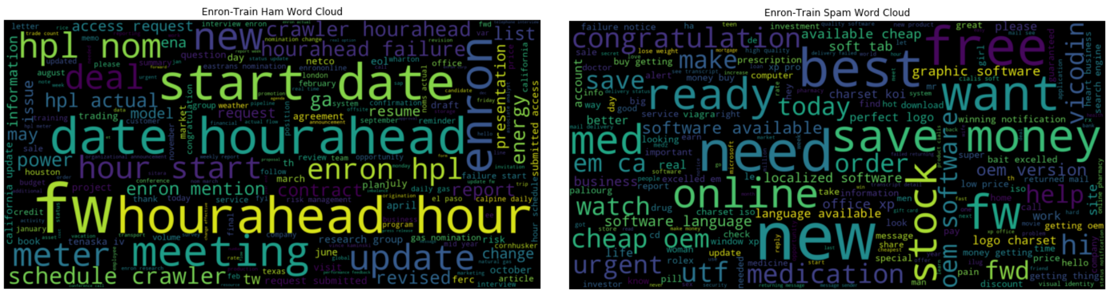


a) b)

Figure 6.30: Ling-spam Subject Field Word Clouds for Test Data

Two datasets were chosen for the word cloud presentation. Since Ling-spam is the smallest dataset of the four. Next Enron dataset word clouds are presented in Figures 6.31 and 6.32.

Ham word cloud for Enron dataset shows words such as ‘hour’, ‘ahead’, ‘date’, ‘meeting in Figure 6.31 a) which is distinctly different to words found in spam word cloud in Figure 6.31 b) for the training samples.



a) b)

Figure 6.31: Enron Subject Field Word clouds for Train Data

In the test samples of Enron, similar distinction in ham and spam words is noticed as shown in Figure 6.32.

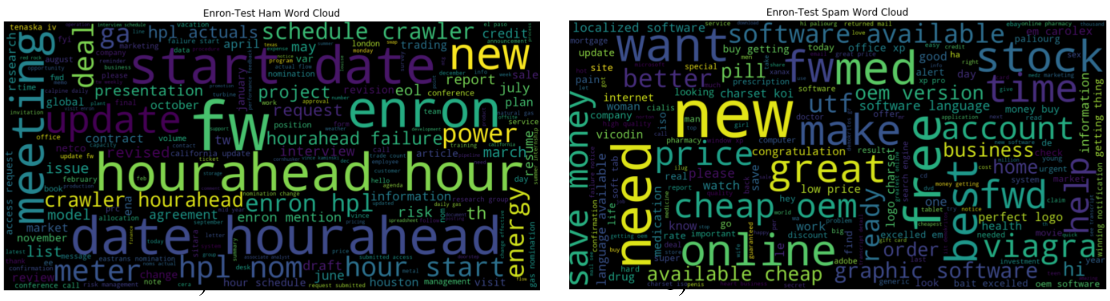


Figure 6.32: Enron Subject Field Word Clouds for Test Data

The performance of the machine learning models logistic regression, support vector machine, XGboost and random forest are recorded via the metrics of accuracy, precision, recall, F1 score, ROC/AUC and PRC curve. The classification reports for each model are compared in Table 6.15. For the Ling-spam dataset, SVM demonstrates outstanding performance with XGboost coming second whereas for the TREC07 dataset the XGBoost outperforms SVM in some metrics.

Table 6.15: Classification Report comparison for Machine Learning Models

Metric->Spam	Accuracy				Precision				Recall				F1 score				ROC/AUC				PRC			
Model ->	LR	SVM	XGB	RF	LR	SVM	XGB	RF	LR	SVM	XGB	RF	LR	SVM	XGB	RF	LR	SVM	XGB	RF	LR	SVM	XGB	RF
Lingspam	91.7	93.3	92	90.5	81	89	86	96	67	69	64	46	73	78	74	62	95.5	94.9	95.8	94.2	81.6	84.1	84.2	83.7
CSDMC2010	87	91.6	90.6	88.3	82	90	89	93	76	83	80	69	79	86	85	79	92.6	95.3	95.6	95.5	88.2	92.9	92.2	91.5
Enron	84.8	92.7	91.9	91.9	84	92	91	91	87	94	93	93	86	93	92	92	92.6	97.3	97.6	97.6	92.8	97.3	97.7	97.7
TREC07	91.9	97.2	97.2	96.2	93	97	97	95	95	98	99	99	94	98	98	97	96.8	98.1	99.4	99.2	96.8	99.5	99.7	99.5

The metrics shown in the confusion matrix support the performance of SVM better than all other models though XGBoost comes close; however, the false positive and false negative rates for XGboost for the TREC07 dataset are higher (Table 6.16) reducing the spam detection performance of the model.

Table 6.16: Confusion Matrix for Machine Learning Models

Metric	FP				FN				TP				TN			
Model	LR	SVM	XGB	RF	LR	SVM	XGB	RF	LR	SVM	XGB	RF	LR	SVM	XGB	RF
Lingspam	22	12	14	3	46	43	50	75	93	96	89	64	663	673	671	682
CSDMC2010	68	36	39	23	99	71	81	126	313	331		286	795	827	824	840
Enron	862	436	463	503	673	303	357	414	4528	4898	4844	4787	4056	4482	4455	4415
TREC07	670	133	285	501	541	77	125	76	9499	4943	9915	9964	4374	2389	4759	4543

Deep learning models, DNN and CNN (version 1 and 2) as explained in Section 6.2 are trained and then evaluated on the test data for all four datasets. The classification reports are shown in Tables 6.17-6.20.

Table 6.17: Classification Report comparison for Deep Learning Models for TREC07 dataset

Metric->	Accuracy	Precision	Recall	F1 score	ROC/AUC	PRC
Model						
DNN	98.4	98	99	99	99	99.5
CNNv1	94.3	92	100	96	98.6	99.2
CNNv2	95.3	94	100	97	98.9	99.3

Table 6.18: Classification Report comparison for Deep Learning Models for Enron dataset

Metric->	Accuracy	Precision	Recall	F1 score	ROC/AUC	PRC
Model						
DNN	91	88	95	92	96.5	96.7
CNNv1	84	78	96	86	93.1	93.2
CNNv2	83	77	96	86	92.7	92.8

Table 6.19: Classification Report comparison for Deep Learning Models for CSDMC2010 dataset

Metric->spam	Accuracy	Precision	Recall	F1 score	ROC/AUC	PRC
DNN	91.4	88	85	86	97	94
CNNv1	87.5	81	80	80	94	89.6
CNNv2	88.2	82	82	82	94	90.4

Table 6.20: Classification Report comparison for Deep Learning Models for Ling-spam dataset

Metric->spam	Accuracy	Precision	Recall	F1 score	ROC/AUC	PRC
DNN	91.3	87	58	69	96.5	85.3
CNNv1	87	64	53	58	92.5	72.3
CNNv2	90	78	55	65	92.5	74

The performance metrics in the results in Tables 6.17-6.20 show that DNN outperforms both versions of CNN for all four datasets. CNN has the best performance for TREC07 dataset. The confusion matrix for all four datasets were recorded and compared.

Table 6.21: Confusion Report comparison for Deep Learning Models for TREC07 dataset

TREC07	FP	FN	TP	TN
DNN	119	78	9962	4879
CNNv1	842	25	10015	4202
CNNv2	677	36	10004	4367

From Table 6.21 it is noticed DNN has the best performance with the least number of FP and FN, indicating that the model has a high spam detection rate.

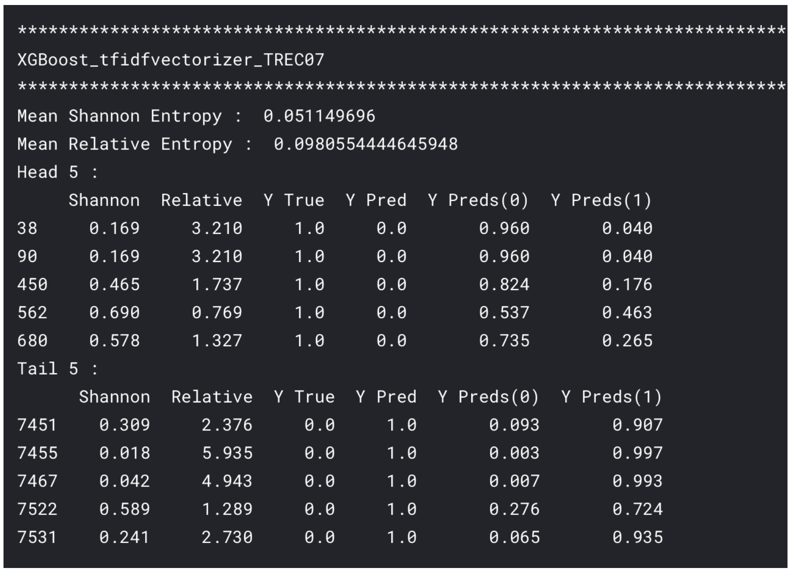
6.6 Discussion:

The performance of entropy methods is evaluated to determine if they can be used to detect email spam when random strings are intentionally injected into email spam. Two entropy types, Shannon and random entropy values were calculated. To determine the entropy values, output probabilities for each sample of the test data is calculated resulting in two main outcomes: Entropy values can validate that the model prediction rate is impacted by introducing random scrambled strings into the emails spam. Relative entropy can be used to detect email spam from ham.

From that, Shannon entropy and Relative entropies for each sample were calculated. The tables in Section 6.5 show that Relative entropy is higher than the Shannon entropy values for all datasets which do not show an identifiable trend before and after adding random strings of data to the dataset used for classification.

These Relative entropy values are then plotted using a histogram chart for example Figure 6.24 to show that the distribution of spam relative entropy (in red) is distinguishable on the graph from ham relative entropy (in green). Next, the Shannon histogram is plotted onto a histogram chart, for example Figure 6.25 to determine that there is no identifiable difference between spam and ham distributions based on Shannon entropy.

From the results, it is evident that incorrect prediction with high probability lead to high relative entropy with confidence whereas the Shannon entropy is not much affected as shown in Figure 6.33.



```
****************************************************************************
XGBoost_tfidfvectorizer_TREC07
****************************************************************************
Mean Shannon Entropy :  0.051149696
Mean Relative Entropy :  0.0980554444645948
Head 5 :
     Shannon  Relative  Y True  Y Pred  Y Preds(0)  Y Preds(1)
38     0.169     3.210     1.0     0.0       0.960       0.040
90     0.169     3.210     1.0     0.0       0.960       0.040
450    0.465     1.737     1.0     0.0       0.824       0.176
562    0.690     0.769     1.0     0.0       0.537       0.463
680    0.578     1.327     1.0     0.0       0.735       0.265
Tail 5 :
      Shannon  Relative  Y True  Y Pred  Y Preds(0)  Y Preds(1)
7451    0.309     2.376     0.0     1.0       0.093       0.907
7455    0.018     5.935     0.0     1.0       0.003       0.997
7467    0.042     4.943     0.0     1.0       0.007       0.993
7522    0.589     1.289     0.0     1.0       0.276       0.724
7531    0.241     2.730     0.0     1.0       0.065       0.935
```

Figure 6.33: Sample Shannon and Relative entropy values with their Prediction Probability for XGBoost model

In Figure 6.33, Shannon and Relative entropy values for the XGboost model applied to the TREC07 dataset are shown which indicate that the higher the probability of incorrect prediction, the high Relative entropy will be. In the Figure, Head 5 means, 5 samples from the top of the spam dataset and Tail 5 means 5 randomly selected samples from the bottom part of the ham dataset. The table underneath shows first column as the sequential number of the selected email, and their entropy values and prediction probability are displayed. Here, Y True is true value (class) of the sample email, Y Pred is the predicted value (class) of the sample email, Y Preds (0) is the probability of the prediction 0 (ham) made by the classifier, also known as confidence and Y Preds (1) is the probability of prediction made by the classifier as 1. For row 38, the true value of the email is spam: however, the model has predicted it as ham with a 0.96 confidence level while the relative entropy for this is 3.210. In contrast, Shannon entropy is 0.169. For row 562, the model again made an incorrect prediction for spam as ham with 0.537 confidence level, and the relative entropy is 0.769. Similarly, for row 7455, a ham email is predicted as spam with a probability of 0.997 and a relative entropy of 5.935 whereas for row 7522, the incorrect prediction is made again with a confidence level of 0.724 and a relative entropy of 1.289. No such trend is noticed for the Shannon entropy. Hence, it is clear that Relative entropy increases if the model makes an erroneous prediction with high confidence.

In order to determine the spam classification performance, histograms showing Shannon and Relative entropies were generated. It is evident that Shannon entropy does not make a clear distinction between ham and spam distributions; however, for each dataset, the entropy values presented in the histograms for ham and spam distribution based on Relative entropy demonstrate that this type of entropy can be used to show classification between spam/ham.

From the research presented in Sections 6.6 and 6.7, an efficient spam filtering technique has been presented based on the subject header of emails. The word vector for the words found in the subject header of emails in said datasets are formed using GloVe embeddings. Models using machine and deep learning algorithms are developed, trained and evaluated on datasets of varying sizes to determine the performance. The dataset sizes range from the smallest dataset Ling-spam (2893) to the largest TREC07(75419) which was selected to determine the comparative performance of deep learning models as these need large dataset to perform at their best. This was evident from the results where both deep learning models produced superior performance for TREC07 dataset. This method of topic-based classification provided the following advantages: 1. Checking only the subject header of the emails in order to increase the executive efficiency and reduce computational complexity. 2. Used a weighted least squared model of embedding for constructing the word vectors as features. 3. Strengthened the ability of email spam detection with this learning mechanism. From the experimental results, precision, recall and F1 score of the proposed method reached 97, 98, and 99 for the best performing model for machine learning and 98, 99, and 99 for the deep learning models. The least scores for Deep learning models were 77, 96, and 86 for CNNv2 using Enron. These performance results for datasets should be ignored due to the relevance of the size of the datasets. The least scores for machine learning models are 86.64 and 74. This means that this method can filter spam effectively without high resource consumption and calculation cost.

With the experimental results in this chapter, the research of thesis concludes. The next chapter will present the conclusion and future work.

Chapter 7

Conclusion and Future Work

In 2004, Bill Gates predicted that "spam will soon be a thing of the past"

7.1 Overview

Email spam has been intimidating users of emails for decades. The increase in the usage of emails has led to, first, emergence and then escalation of emails as spam used for phishing, hacking, malware spread, identity theft, scams related to online shopping, dating, remote access investment, income tax and threats to life. In recent years, email spam filtering tools have become a necessity for internet service providers and organisations to tackle the phenomenon of continuous increase in sophistication of email spam. Filtering tools are made up of one or more modules focused on detecting different features of such spam. The most common filters investigated by researchers rely on list-based, rule-based or text categorisation techniques for analysis of semantic features of emails – using one or a combination of features.

Significant research effort has gone into this area with considerable success; nonetheless. the problem of email spam persists, despite numerous existing methods to detect email spam. The volume of undetected spam has continued to stay consistent at unacceptable levels and these intrusive and dangerous messages continue to land in user inboxes. Thus, an optimal solution has yet to be created, especially that each user's needs are different. Planting an email spam filter at the server-side offers a 'one size fits all' solution which does not apply to individual user as user email spam is in the 'eye of the beholder'. To protect a user's inbox from email spam and offer a customised solution, a client-side filter is required.

For complete protection from the threat email spam poses, a customised filter needs to be developed for a user, which is difficult. Even with such measures, it is still not possible to guarantee an inbox free of email spam as vulnerabilities exist and spammers keep developing with new ways to circumvent some processes. With significant input from research studies, content-based filters are believed to provide a practical solution for identifying email spam. Such filters develop a statistical model of ham email characteristics from a set of training samples. If new email characteristics deviate significantly from the pre-developed model, then it generates an alarm and classifies the email as spam. However, existing content filtering approaches suffer from significant weaknesses as email spam contains attributes beyond the text content and spammers play with textual features to confuse filters. Hence, an inbox protected by a content-based filter dependent only on text features cannot completely eliminate all email spam.

This thesis highlights a need for further refinement of the approaches used for spam control. Most importantly, the thesis identifies and implements techniques to neutralize spammers' techniques. In this chapter, the thesis is summarized in Section 7.2, contributions are reviewed in Section 7.3, and possible future work is explored in Section 7.4.

7.2 Summary

Email spam has been the focus of studies for decades. Though there are many different techniques to block spam email messages from reaching users' inboxes, filtering is the most commonly used mechanism and has had some success levels of email spam are still unacceptably high. Researchers and organisations create filters that are smart and self-learning, but spammers are generally one step ahead. They continue to develop new techniques to deceive filters and their learning mechanisms. Hence, the problem still remains and with it scope for research in this area.

In this thesis, the problem of detecting email spam at the client-side is addressed. In particular, novel frameworks are introduced and models are developed which address critical issues that severely impact the success rate for email spam detection. These issues are

- Limited coverage of email spam detection through server-side filtering
- Variations in user preferences for email spam
- Limitation of text only features
- Large number of false alarms as FP and FN
- Changing spammer techniques

This current work is an effort in the same field to reduce false negatives/spam in the inbox of users which has deceived organisational filters. This research applies further filtering by training the filter with user specific data which, as results demonstrate is effective in reducing the volume of false positives.

As a base point, in Chapter 3, we tested the performance of a Bayesian classifier with a range of parameters. These included different settings for thresholds and token sizes as well as characteristics of feature sets such as unigrams, bigrams and trigrams of different sizes ranging from 75 to 20000; it was noted that there is room for improvement. The Bayesian algorithm classifies a new email as spam, ham or grey. Greys are isolated in an area that the user has to manually classify to eliminate the false positives and false negatives. Results demonstrated that, when parameters are optimised, the least amount of FP and FN occurs, but the volume of greys still needs to be reduced.

Email spam is annoying and causes financial loss to organizations and individual users. This thesis in chapter 4, focused on improving classification rates using semantic and syntactic features, bigrams, improved feature selection methods and a supervised machine learning-based ensemble. The performance of the feature selection techniques and implementation was presented. A dynamic multi-layer ensemble model (DMLEM) was proposed to eliminate greys from the Bayesian classifier and tested on 10 datasets; results showed that 99+% correct classification was achieved, with FP as low as 0%, the highest being 0.9% and averaging 0.3-0.4%.

In chapter 5, it was hypothesised that user profiling plays an impotent role in email spam classification. User profiles were developed using significant phrases identified from the content of emails as part of feature extraction. Feature selection was performed using three methods to determine optimal performance and evaluated using machine learning and deep learning techniques. Overall, machine learning techniques resulted in higher performance compared to deep learning methods. However, the performance of deep learning algorithms was high. Results showed competitive performance by the three methods used for feature selection. It was proven that precision of phrase-based filtering is high; this has been evaluated through several classification algorithms and thus can be considered reliable for user profiling for email classification and is more accurate than just keyword-based models.

To confirm that information categorisation based on user preferences is required for email spam filtering, several experiments were conducted for all user data using machine learning and deep learning methods. Depending on user preferences, it was possible to significantly reduce the error rate while filtering emails for different users.

As future work for user profiling, clustering needs to be investigated to determine the impact on categorisation and user profiling to increase the classification detection rate and accuracy of user preferences.

It is challenging to maintain the efficiency of the email spam classification filters even without constantly changing spammer methods which complicate this process further. A recently introduced spamming method is the injection of random valid strings into email spam to deceive the filters. Such injection poisons the feature sets used to learn the filters with respect to spam and ham emails. This poses a significant challenge to architects of email spam classification filters as this increasing language variety in email spam makes it difficult to build filters with acceptable performance and reliability.

Hence, chapter 6 introduced methods that contribute to the robustness of email spam detection. First, a new entropy-based email spam detection method was proposed for the case of random string injection and for efficiently detecting email spam where random valid text is injected into the spam emails. A series of experiments were carried out for identifying entropy values before and after the addition of valid random strings and again after scrambled random strings were injected into email spam. The results demonstrated that the presence of such random valid strings does impact the detection capability of email spam filters and that Relative entropy can successfully differentiate spam from ham emails.

Second, attention has been given to the subject field of the emails. Recently, spammers have focused on the topics of email spam to lure users to read and respond to those emails. In this chapter, topic-based email spam classification is proposed. The results show that this classification method is a low cost, high detection mechanism to classify email spam.

7.3 Thesis Contribution

This thesis contribution involved the development of a recursive multistage classifier for email spam classifier at the client-side (AREMUCCS) with low computation cost and constitutes of the following stages:

- Identified and addressed critical issues that impact the successful classification of email spam for a user and email spam detection filters at the server-side.
- Developed a user profiling system that models user profiles statistically based on keyword phrases in the user data and collects individual user preferences. The effectiveness based on such profiles has been demonstrated using machine learning and deep learning models which showed that email classification performance was high using Enron datasets containing email ham and spam from real users.
- Introduced a Bayesian Classifier that acts as another layer of filtering (stage 2) at the client-side to 1. Provide user specific filtering with a context specific dataset to filter emails into three categories - ham, spam and grey 2. refilter email spam classified by server-side filters to identify false alarms.
- Developed a novel, proficient multi-layer (stage 3) dynamic ensemble model based on a bagging technique using BoW, term frequency document and chi square approaches to extract, select and order features for email feature selection. This model incorporates novel semantic and syntactic features that include the structural characteristics of an email to distinguish ham and spam emails for a user at the email client.
- Identified a new spammer method of random string injection to deceive the learning-based filters and developed an efficient entropy-based email spam detection system to detect injected random valid strings as well as strings that are injected scrambled to confuse filters. The

proposed model has been implemented using machine and deep learning algorithms. The models have been evaluated using TREC07, Enron, CSDMC2010 and Ling-spam datasets.

- Developed an efficient, low computation cost and precise model for email spam detection based on the topic of emails using GloVE embeddings for feature selection. The topic-based email spam detection method is capable of discriminating between spam and ham in real time.

7.4 Future Work

Studies reported by Nucleus research in 2020 indicate that the cost to organisations of email spam per employee is $1934.00, and predictions are not promising in relation to future cost of email spam. In studies conducted by the Radicati Research Group Inc., a research firm based in Palo Alto, California, spam costs businesses $20.5 billion annually in implementing technical solutions and in decreased productivity. Spam Laws suggest that if spam continues to grow at its current rate, future cost to businesses may be far more substantial (Laws, 2020).

This confirms that even though email spam detection has been the focus of research for more than a decade, it is not going to lose traction anytime soon. Hence, further contribution to this area is inevitable. Some interesting avenues for future research are:

- User profiling has been developed using user data provided by Enron dataset; this method needs to be evaluated in real time with real users.
- Although the DMLEM classifier is an 'email' spam detection system, it has, through its syntactic features, the capability to effectively also detect 'review' spam (a different type of spam).
- The stages in AREMUCCS have been evaluated using four datasets; to further test the effectiveness and performance of this classifier, it would be useful to test it with real users.
- Different stages of the AREMUCCS classifier lends itself to consolidation into one single software application that can be installed on the email client of a user.
- Topic-based classification using GloVe embeddings is an email spam detection system; it can be extended to other types of spam such as web-based spam attacks or industry-specific spam.
- Deep learning models such as DNN and CNN have been trained and evaluated with large datasets such as TREC07 which contains 75,000 emails. This needs to be tested with even larger datasets to make the best use of deep learning methods and measure their effectiveness for email spam detection.
- The number of datasets containing email spam and ham is limited and the latest available is from 2015. As future work, development of email spam and ham dataset would be useful to help the research community.

Bibliography:

Abu-Nimeh, S., Nappa, D., Xinlei, W., & Nair, S. (2008). Bayesian Additive Regression Trees-Based Spam Detection for Enhanced Email Privacy. In (pp. 1044-1051).

Aggarwal, C. Z., ChengXiang. (2012). A Survey of Text Classification Algorithms. In C. C. Aggarwal & C. Zhai (Eds.), *Mining Text Data* (pp. 163-222): Springer US.

Akinyelu, A., & Adewumi, A. (2014). Classification of Phishing Email Using Random Forest Machine Learning Technique. *Journal of Applied Mathematics, 2014*(2014). doi:10.1155/2014/425731

Aldwairi, M., & Flaifel, Y. (2012, 18-20 Sept. 2012). *Baeza-Yates and Navarro approximate string matching for spam filtering.* Paper presented at the Innovative Computing Technology (INTECH), 2012 Second International Conference on.

Ali, A. B. M. S., & Xiang, Y. (2007). *Spam Classification Using Adaptive Boosting Algorithm.* Paper presented at the Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on.

Ali Elsiddig, A. O., Elhadi, A. A. E., & Ahmed, A. (2017). Features Reweighting and Similarity Coefficient Based Method for Email Spam Filtering. *American Journal of Applied Sciences, 14*(10), 983-993. doi:10.3844/ajassp.2017.983.993

Ali, S. (2019). Spam image email filtering using K-NN and SVM. *International Journal of Electrical and Computer Engineering, 9*(1), 245-254. doi:10.11591/ijece.v9i1.pp245-254

Alsowail, M., & Batarfi, O. (2011). Filtering spam emails based on user behaviors. *International Journal of Advanced Research in Computer Science, 2*(5).

Anandita, Yadav, D. P., Paliwal, P., Kumar, D., & Tripathi, R. (2017). *A Novel Ensemble Based Identification of Phishing E-Mails*. Paper presented at the Proceedings of the 9th International Conference on Machine Learning and Computing, Singapore, Singapore.

Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C., & Stamatopoulos, P. (2000). Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach. *arXiv.org*.

Androutsopoulos, I., Paliouras, G., & Michelakis, E. (2006). *Learning to Filter Unsolicited Commercial E-Mail*.

Androutsopoulos, J. K., Chandrinos, K.V. Paliouras,G and Spyropoulos, C.D. . (2000a). *An Evaluation of Naive Bayesian Anti-Spam Filtering.* Paper presented at the 11th European Conference on Machine Learning, Barcelona, Spain.

Androutsopoulos, J. K., Chandrinos, K.V. Paliouras,G and Spyropoulos, C.D. . (2000b). *An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Personal E-mail Messages.* Paper presented at the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR2000), Athens, Greece.

Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., & Feamster, N. (2010). *Building a dynamic reputation system for DNS*. Paper presented at the Proceedings of the 19th USENIX conference on Security, Washington, DC.

Aradhye, H. B., Myers, G. K., & Herson, J. A. (2005, 29 Aug.-1 Sept. 2005). *Image analysis for efficient categorization of image-based spam e-mail.* Paper presented at the Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on.

Bai, S., Kolter, J., & Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.

Bajaj, K. (2017). *A Multi-layer Model to Detect Spam Email at Client Side*, Cham.

Bajaj, K., & Pieprzyk, J. (2014). *A case study of user-level spam filtering*. Paper presented at the Proceedings of the Twelfth Australasian Information Security Conference - Volume 149, Auckland, New Zealand.

Bajaj, K. S. (2016). *A Multi-layer Model to Detect Spam Email at Client Side*. Paper presented at the SecureComm 2016, China.

Bajaj, S. K., & Pieprzyk, J. (2013, 21-22 Nov. 2013). *Can We CAN the Email Spam.* Paper presented at the Cybercrime and Trustworthy Computing Workshop (CTC), 2013 Fourth.

Balakumar, M., & Vaidehi, V. (2008, 4-6 Jan. 2008). *Ontology based classification and categorization of email.* Paper presented at the Signal Processing, Communications and Networking, 2008. ICSCN '08. International Conference on.

Barushka, A., & Hajek, P. (2018). Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. *Applied Intelligence*. doi:10.1007/s10489-018-1161-y

Beigy, M. F. S. a. H. (2012). Learning to filter spam emails: An ensemble learning approach. *International Journal of Hybrid Intelligent Systems, 9*, 27-43. doi:10.3233/HIS-2011-0145

Bhowmick, A., & Hazarika, S. M. (2018, 2018//). *E-Mail Spam Filtering: A Review of Techniques and Trends.* Paper presented at the Advances in Electronics, Communication and Computing, Singapore.

Blanzieri, E., & Bryl, A. (2008). A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review, 29*(1), 63-92. doi:10.1007/s10462-009-9109-6

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5-32. doi:10.1023/a:1010933404324

Brewer, D., Thirumalai, S., Gomadam, K., & Kang Li, A. K. L. (2006). *Towards an Ontology Driven Spam Filter.* Paper presented at the Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on.

Brownie, J. (2018). A Data-Driven Approach to Choosing Machine Learning Algorithms. *Start Machine Learning*. Retrieved from https://machinelearningmastery.com/a-data-driven-approach-to-machine-learning/

Brownlee, J. (2016). A Gentle Introduction to XGBoost for Applied Machine Learning. *XGBoost*. Retrieved from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Brun, O., Yin, Y., & Gelenbe, E. (2018). Deep Learning with Dense Random Neural Network for Detecting Attacks against IoT-connected Home Environments. In (Vol. 134, pp. 458-463).

Caruana, G., & Li, M. (2008). A survey of emerging approaches to spam filtering. *ACM Comput. Surv., 44*(2), 1-27. doi:10.1145/2089125.2089129

Caruana, G., & Li, M. (2012). A survey of emerging approaches to spam filtering. *ACM Comput. Surv., 44*(2), 1-27. doi:10.1145/2089125.2089129

Caruana, G., Maozhen, L., & Man, Q. (2011, 26-28 July 2011). *A MapReduce based parallel SVM for large scale spam filtering.* Paper presented at the Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on.

Chakrabarty, A., & Roy, S. (2014). An optimized k-NN classifier based on minimum spanning tree for email filtering. In (pp. 47-52).

Chao, X., & Yiming, Z. (2007, 15-19 Dec. 2007). *Transductive Support Vector Machine for Personal Inboxes Spam Categorization.* Paper presented at the Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In (Vol. 13-17-, pp. 785-794).

Cheng, V., & Li, C. h. (2006, 18-22 Dec. 2006). *Personalized Spam Filtering with Semi-supervised Classifier Ensemble.* Paper presented at the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06).

Chhabra, S., Yerazunis, W. S., & Siefkes, C. (2004). *Spam filtering using a Markov random field model with variable weighting schemas.* Paper presented at the Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on.

Chharia, A., & Gupta, R. K. (2013, 8-10 Aug. 2013). *Email classifier: An ensemble using probability and rules.* Paper presented at the 2013 Sixth International Conference on Contemporary Computing (IC3).

Chih-Chin, L., & Ming-Chi, T. (2004, 5-8 Dec. 2004). *An empirical performance comparison of machine learning methods for spam e-mail categorization.* Paper presented at the Hybrid Intelligent Systems, 2004. HIS '04. Fourth International Conference on.

Chinavle, D., Kolari, P., Oates, T., & Finin, T. (2009). *Ensembles in adversarial classification for spam*. Paper presented at the Proceedings of the 18th ACM conference on Information and knowledge management, Hong Kong, China.

Ching-Tung, W., Kwang-Ting, C., Qiang, Z., & Yi-Leh, W. (2005, 11-14 Sept. 2005). *Using visual features for anti-spam filtering.* Paper presented at the IEEE International Conference on Image Processing 2005.

Clement, J. (2019). Spam: share of global email traffic 2014-2019. *Cyber Crime/Statistics*. Retrieved from https://www.statista.com/statistics/420391/spam-email-traffic-share/

Clifford, M., Faigin, D., Bishop, M., & Brutch, T. (2003). *Miracle Cures and Toner Cartridges: Finding Solutions to the Spam Problem.* Paper presented at the 19th annual computer security applications conference (ACSAC 2003).

Cormack G V., L. R. T. (2007). *2007 TREC Public Spam Corpus*. TREC07. Retrieved from: https://plg.uwaterloo.ca/~gvcormac/treccorpus07/about.html

Cormack, G. V., & Cruz, J.-M. M. d. (2009). *On the relative age of spam and ham training samples for email filtering*. Paper presented at the Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, Boston, MA, USA.

Cormack, G. V., & Lynam, T. R. (2007). Online supervised spam filter evaluation. *ACM Trans. Inf. Syst., 25*(3), 11. doi:10.1145/1247715.1247717

Cristianini, Nello, & Shawe-Taylor, J. (2001). An introduction to support vector machines and other kernel-based learning methods. Repr. *Introduction to Support Vector Machines and other Kernel-Based Learning Methods, 22*. doi:10.1017/CBO9780511801389

Dada, E., & Joseph, S. (2018a). Logistic Model Tree Induction Machine Learning Technique for Email Spam Filtering. *19*, 96-102.

Dada, E., & Joseph, S. (2018b). *Random Forests Machine Learning Technique for Email Spam Filtering*.

Dada, E., Joseph, S., Chiroma, H., Abdulhamid, S. i., Adetunmbi, A., Opeyemi, E., & Ajibuwa. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon, 5*, 1-23. doi:10.1016/j.heliyon.2019.e01802

De, W., Irani, D., & Pu, C. (2013). A study on evolution of email spam over fifteen years. In (pp. 1-10).

Deepak, P., & Sandeep, P. (2005). *Spam filtering using spam mail communities.* Paper presented at the Applications and the Internet, 2005. Proceedings. The 2005 Symposium on.

Deffree, S. (2019). 1st spam email is sent, May 3, 1978. *EDN Moments*. Retrieved from https://www.edn.com/1st-spam-email-is-sent-may-3-1978/

Delany, S. J., Cunningham, P., Tsymbal, A., & Coyle, L. (2005). A Case-Based Technique for Tracking Concept Drift in Spam Filtering. In *Applications and Innovations in Intelligent Systems XII* (pp. 3-16).

Denning, P. (1982). ACM president's letter: electronic junk. *Communications of the ACM, 25*(3), 163-165. doi:10.1145/358453.358454

Deshpande, V. P., Erbacher, R. F., & Harris, C. (2007, 20-22 June 2007). *An Evaluation of Naïve Bayesian Anti-Spam Filtering Techniques.* Paper presented at the 2007 IEEE SMC Information Assurance and Security Workshop.

Devi, K. (2018). Random Forests Spam Email Classification System. *Journal of Computer Engineering & Information Technology, 07*(1). doi:10.4172/2324-9307.1000190

Dhinakaran, C., Chae, C.-J., & Lee, J.-K. (2007). *An Empirical Study of Spam and Spam Vulnerable email Accounts.* Paper presented at the Future generation communication and networking (fgcn 2007).

Diale, M., Celik, T., & Van Der Walt, C. (2019). Unsupervised feature learning for spam email filtering. *Computers & Electrical Engineering, 74*, 89-104. doi:https://doi.org/10.1016/j.compeleceng.2019.01.004

Drucker, H., Donghui, W., & Vapnik, V. N. (1999). Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on, 10*(5), 1048-1054.

Du, L., Song, Q., & Jia, X. L. (2014). Detecting concept drift: An information entropy based method using an adaptive sliding window. *Intelligent Data Analysis, 18*(3), 337-364. doi:10.3233/IDA-140645

Du, M., Yu, Q., Fei, S., Wang, C., Gong, X., & Luo, R. (2019). *Fully Dense Neural Network for the Automatic Modulation Recognition*.

du Toit, T., & Kruger, H. (2012, 15-17 Aug. 2012). *Filtering spam e-mail with Generalized Additive Neural Networks.* Paper presented at the Information Security for South Africa (ISSA), 2012.

Esquivel, H., Akella, A., & Mori, T. (2010). On the effectiveness of IP reputation for spam filtering. In (pp. 1-10).

Fahad, S. (2015). Developing a spam Email Detector.

Fdez-Riverola, F., Iglesias, E. L., DÃaz, F., MÃ©ndez, J. R., & Corchado, J. M. (2007). SpamHunting: An instance-based reasoning system for spam labelling and filtering. *Decision Support Systems, 43*(3), 722-736. Retrieved from riverola@uvigo.es

10.1016/j.dss.2006.11.012

http://search.ebscohost.com/login.aspx?direct=true&db=psyh&AN=2007-05229-004&site=ehost-live

Fdez-Riverola, F., Iglesias, E. L., Díaz, F., Méndez, J. R., & Corchado, J. M. (2007). Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications, 33*(1), 36-48. doi:10.1016/j.eswa.2006.04.011

Ferris_Research. (2007). Spam Control: The Current Landscape. Retrieved from http://www.ferris.com/2007/01/02/the-commodity-status-of-spam-control/

Firte, L., Lemnaru, C., & Potolea, R. (2010). *Spam detection filter using KNN algorithm and resampling*.

Flach, P. (2019). Performance Evaluation in Machine Learning: The Good, the Bad, the Ugly, and the Way Forward. *Proceedings of the AAAI Conference on Artificial Intelligence, 33*, 9808-9814. doi:10.1609/aaai.v33i01.33019808

Ford, R., & Spattord, E. H. (2007). Happy Birthday, Dear Viruses. *Science, 317*(5835), 210-211. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=pbh&AN=25917610&site=ehost-live

Garg, A., Battiti, R., & Cascella, R. G. (2006). *"May I borrow your filter?" Exchanging filters to combat spam in a community.* Paper presented at the Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on.

Gargiulo, F., Penta, A., Picariello, A., & Sansone, C. (2009). A Personal Antispam System Based on a Behaviour-Knowledge Space Approach. In O. Okun & G. Valentini (Eds.), *Applications of Supervised and Unsupervised Ensemble Methods* (pp. 39-57). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gashti, M. Z. (2017). Detection of Spam Email by Combining Harmony Search Algorithm and Decision Tree. *Engineering, Technology & Applied Science Research, 7*(3), 1713-1718. doi:10.5281/zenodo.809513

George, P., & Vinod, P. (2015). *Machine learning approach for filtering spam emails*. Paper presented at the Proceedings of the 8th International Conference on Security of Information and Networks, Sochi, Russia.

Goel, E., Abhilasha. (2017). Random Forest: A Review. *International Journal of Advanced Research in Computer Science and Software Engineering, 7*(1), 251-257. doi: 10.23956/ijarcsse/V7I1/01113

Gomez, J., & Moens, M.-F. (2010). Using Biased Discriminant Analysis for Email Filtering. In R. Setchi, I. Jordanov, R. Howlett, & L. Jain (Eds.), *Knowledge-Based and Intelligent Information and Engineering Systems* (Vol. 6276, pp. 566-575): Springer Berlin Heidelberg.

Graham, P. (2002). A Plan for Spam. Retrieved from http://www.paulgraham.com/spam.html

Grimes, G. A. (2007). Compliance with the CAN-SPAM Act of 2003. *Commun. ACM, 50*(2), 56–62. doi:10.1145/1216016.1216021

Grobelnik, M., Mladenić, D., & Fortuna, B. (2009). Ontology Generation from Social Networks. In J. Davies, M. Grobelnik, & D. Mladenić (Eds.), *Semantic Knowledge Management* (pp. 129-139): Springer Berlin Heidelberg.

Gudkova, D., & Namestnikova, M. (2011). Spam in the First Quarter of 2011. (April 2011). Retrieved from http://www.securelist.com/en/analysis/204792175/Spam_in_the_First_Quarter_of_2011

Guoqing, M., Wei, Z., Haixia, C., & Jianshe Dong, A. J. D. (2006). *Multi-agent Interaction Based Collaborative P2P System for Fighting Spam.* Paper presented at the

Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on.

Guoyang, S., Bin, G., Tie-Yan, L., Guang Feng, A. G. F., Shiji Song, A. S. S., & Hang Li, A. H. L. (2006). *Detecting Link Spam Using Temporal Information.* Paper presented at the Data Mining, 2006. ICDM '06. Sixth International Conference on.

Gupta, R., Kumar, K. V., & Mohandas, R. (2011, 22-24 April 2011). *Spam control by source throttling using integer factorization.* Paper presented at the 2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC).

Guyon, I., & Elisseeff, A. (2003). An Introduction of Variable and Feature Selection. *J. Machine Learning Research Special Issue on Variable and Feature Selection, 3*, 1157-1182. doi:10.1162/153244303322753616

Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to Spam filtering. *Expert Systems with Applications, 36*(7), 10206-10222. doi:10.1016/j.eswa.2009.02.037

Gyöngyi, Zoltán, Garcia, M., & Hector, H. (2005). *Web Spam Taxonomy*.

Hajara, M., Gital, A. Y., Zambuk, F. U., Umar, A., Umar, A. Y., & Waziri, J. U. (2019). A comparative analysis of phishing website detection using XGBOOST algorithm. *Journal of Theoretical and Applied Information Technology, 97*, 1434-1443.

Han, A., Kim, H., Ha, I., & Jo, G. (2008, 10-11 July 2008). *Semantic Analysis of User Behaviors for Detecting Spam Mail.* Paper presented at the 2008 IEEE International Workshop on Semantic Computing and Applications.

Harisinghaney, A., Dixit, A., Gupta, S., & Arora, A. (2014, 6-8 Feb. 2014). *Text and image based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm.* Paper presented at the 2014 International Conference on Reliability Optimization and Information Technology (ICROIT).

Hazim, M., Anuar, N. B., Ab Razak, M. F., & Abdullah, N. A. (2018). Detecting opinion spams through supervised boosting approach. *PLOS ONE, 13*(6), e0198884. doi:10.1371/journal.pone.0198884

Ho, C. C., Baharim, K. N., Abdulsalam, A., & Alias, M. S. (2017). *Deep Neural Networks for Text: A Review*.

Hoanca, B. (2006). How good are our weapons in the spam wars? *Technology and Society Magazine, IEEE, 25*(1), 22-30.

Horie, M., & Neville, S. W. (2008). Addressing Spam at the Systems-level through a Peered Overlay Network-Based Approach. In *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics* (pp. 449-453).

Huai-bin, W., Ying, Y., & Zhen, L. (2005). SVM Classifier Incorporating Feature Selection Using GA for Spam Detection. In *Embedded and Ubiquitous Computing* (pp. 1147-1154).

Huang, G., & Xu, Y. (2013). Hybrid spam filtering method based on users' feedback. *Jisuanji Yingyong / Journal of Computer Applications, 33*(7), 1861-1865. doi:10.11772/j.issn.1001-9081.2013.07.1861

Iqbal, M., Muneeb Abid, M., Ahmad, M., & Khurshid, F. (2016). Study on the Effectiveness of Spam Detection Technologies. *International Journal of Information Technology and Computer Science, 8*, 11-21. doi:10.5815/ijitcs.2016.01.02

Iqbal, M., Shoukat, A., Khan, S., & Iqbal, S. (2011). Performance Analysis of K-NN and Naïve Bayes Classifiers for Spam Filtering Application. *International Journal of Advanced Research in Computer Science, 2*(2).

Isacenkova, J., & Balzarotti, D. (2011). *Measurement and Evaluation of a Real World Deployment of a Challenge-Response Spam Filter*.

Isacenkova, J., & Balzarotti, D. (2014). *Shades of gray: a closer look at emails in the gray area*. Paper presented at the Proceedings of the 9th ACM symposium on Information, computer and communications security, Kyoto, Japan.

Islam, M., & Zhou, W. (2007). Architecture of Adaptive Spam Filtering Based on Machine Learning Algorithms

Algorithms and Architectures for Parallel Processing. In H. Jin, O. Rana, Y. Pan, & V. Prasanna (Eds.), (Vol. 4494, pp. 458-469): Springer Berlin / Heidelberg.

Islam, M. R., Wanlei, Z., & Chowdhury, M. U. (2008, 14-16 May 2008). *Email Categorization Using (2+1)-Tier Classification Algorithms.* Paper presented at the Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on.

Islam, R., & Wanlei, Z. (2007, 3-6 Dec. 2007). *Email Categorization Using Multi-stage Classification Technique.* Paper presented at the Parallel and Distributed Computing, Applications and Technologies, 2007. PDCAT '07. Eighth International Conference on.

Jacovi, A., Sar Shalom, O., & Goldberg, Y. (2018). *Understanding Convolutional Neural Networks for Text Classification*.

Jacovi, A., Shalom, O., & Goldberg, Y. (2018). *Understanding Convolutional Neural Networks for Text Classification*.

Jaeyeon, J., & Emil, S. (2004). *An empirical study of spam traffic and the use of DNS black lists*. Paper presented at the Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy.

Jiang, F., Jin, G., Yuanyuan, X., & Guandong, X. (2016). Coupled behavioral analysis for user preference-based email spamming. In (pp. 1-5).

Jindal, N., & Liu, B. (2007). *Analyzing and Detecting Review Spam.* Paper presented at the Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on.

Junejo, K., & Karim, A. (2013). Robust personalizable spam filtering via local and global discrimination modeling. *Knowledge and Information Systems, 34*(2), 299-334. doi:10.1007/s10115-012-0477-x

Junejo, K. N., & Karim, A. (2013). *Robust personalizable spam filtering via local and global discrimination modeling.*

Kala, R. (2016). 3 - Perception in Autonomous Vehicles. In R. Kala (Ed.), *On-Road Intelligent Vehicles* (pp. 36-58): Butterworth-Heinemann.

Karthika Renuka, D., Hamsapriya, T., Raja Chakkaravarthi, M., & Lakshmi Surya, P. (2011, 20-22 July 2011). *Spam Classification Based on Supervised Learning Using Machine Learning Techniques.* Paper presented at the Process Automation, Control and Computing (PACC), 2011 International Conference on.

Khamis, S., Mohd Foozy, C. F., Aziz, M., & Rahim, N. (2020). Header Based Email Spam Detection Framework Using Support Vector Machine (SVM) Technique. In (pp. 57-65).

Khater, I. (2012). *Identifying Potentially Useful Email Header Features for Email Spam Filtering*. Paper presented at the CDS 2012 : The Sixth International Conference on Digital Society, Valencia, Spain.

Kholghi, R., Roudsari, S. B., & Pour, A. N. (2011). An efficient spam mail detection by counter technique. *World Academy of Science, Engineering and Technology, 74*, 579-582.

Kigerl, A. (2009). CAN SPAM Act: An Empirical analysis. *International Journal of Cyber Criminology (IJCC), 3*, 974-2891.

Kim, H.-J., Shrestha, J., Kim, H.-N., & Jo, G.-S. (2006). User Action Based Adaptive Learning with Weighted Bayesian Classification for Filtering Spam Mail. In A. Sattar & B.-h. Kang (Eds.), *AI 2006: Advances in Artificial Intelligence* (Vol. 4304, pp. 790-798): Springer Berlin Heidelberg.

Kim, J., Chung, K., & Choi, K. (2007). Spam Filtering With Dynamically Updated URL Statistics. *IEEE Security & Privacy, 5*(4), 33-39. doi:10.1109/MSP.2007.95

Kim, J., Dou, D., Liu, H., & Kwak, D. (2007). Constructing a User Preference Ontology for Anti-spam Mail Systems. In Z. Kobti & D. Wu (Eds.), *Advances in Artificial Intelligence* (Vol. 4509, pp. 272-283): Springer Berlin Heidelberg.

Klimt, B., & Yang, Y. (2004). *Introducing the Enron Corpus*.

Klonowski, M., & Strumiński, T. (2008). Proofs of Communication and Its Application for Fighting Spam. In *SOFSEM 2008: Theory and Practice of Computer Science* (pp. 720-730).

Kolcz, A., Bond, M., & Sargent, J. (2006). *The challenges of service-side personalized spam filtering: scalability and beyond*. Paper presented at the Proceedings of the 1st international conference on Scalable information systems, Hong Kong. https://doi-org.ezproxy.uws.edu.au/10.1145/1146847.1146868

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems, 25*. doi:10.1145/3065386

Kuipers, B. J., Liu, A. X., Gautam, A., & Gouda, M. G. (2005). Zmail: zero-sum free market control of spam. In (pp. 20-26).

Kun-Lun, L., Kai, L., Hou-Kuan, H., & Sheng-Feng, T. (2002, 2002). *Active learning with simplified SVMs for spam categorization.* Paper presented at the Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on.

Lai, C.-C. (2007). An empirical study of three machine learning methods for spam filtering. *Knowledge-Based Systems, 20*(3), 249-254. Retrieved from cclai@mail.nutn.edu.tw

10.1016/j.knosys.2006.05.016

http://search.ebscohost.com/login.aspx?direct=true&db=psyh&AN=2007-05169-003&site=ehost-live

Laws, S. (2020). Spam Statistics and Facts. Retrieved from https://www.spamlaws.com/spam-stats.html

Leavitt, N. (2007). Vendors Fight Spam's Sudden Rise. *Computer, 40*(3), 16-19.

Levenstein, J. (2013). *Email statistics report, 2013-2017*. Retrieved from 1900 EMBARCADERO ROAD, SUITE 206. • PALO ALTO, CA 94303: http://www.radicati.com/?p=9659

Levine, J. (2005). *Experiences with Greylisting*.

Li, Z., & Shen, H. (2011, 10-15 April 2011). *SOAP: A Social network Aided Personalized and effective spam filter to clean your e-mail box.* Paper presented at the 2011 Proceedings IEEE INFOCOM.

Liang, T., & Yu, Q. (2012, 2-4 Nov. 2012). *Spam Feature Selection Based on the Improved Mutual Information Algorithm.* Paper presented at the Multimedia Information Networking and Security (MINES), 2012 Fourth International Conference on.

Liu, P., Dong, J.-s., & Zhao, W. (2007). A Statistical Spam Filtering Scheme Based on Grid Platform. In *Theoretical Advances and Applications of Fuzzy Logic and Soft Computing* (pp. 527-534).

Liu, X., Zou, P., Zhang, W., Zhou, J., Dai, C., Wang, F., & Zhang, X. (2017). CPSFS: A Credible Personalized Spam Filtering Scheme by Crowdsourcing. *Wireless Communications and Mobile Computing, 2017*, 1-9. doi:10.1155/2017/1457870

Lowd, D., & Meek, C. (2005). *Good Word Attacks on Statistical Spam Filters*.

Maria Vergelis, T. S., Tatyana Shcherbakova. (2019). Spam and phishing in Q3 2019. *SPAM AND PHISHING REPORTS, Q3, 2019*(Q3, 2019). Retrieved from https://securelist.com/spam-report-q3-2019/95177/

Marsland, S. (2014). *Machine Learning : An Algorithmic Perspective, Second Edition*. Bosa Roca: Bosa Roca: CRC Press LLC.

McDowell, M. (2006). *Defending Cell Phones and PDAs Against Attack*. USA: National Cyber Alert System Retrieved from http://www.us-cert.gov/cas/tips/ST06-007.html

McGibney, J., & Botvich, D. (2007). Establishing Trust Between Mail Servers to Improve Spam Filtering. In *Autonomic and Trusted Computing* (pp. 146-155).

Meyer , T. A., & Whateley, B. (2004). *SpamBayes: Effective open-source, Bayesian based, email classification system.* . Paper presented at the First Conference on Email and Anti-Spam (CEAS) Mountain View, CA

Ming, L., Yunchun, L., & Wei, L. (2007). *Spam Filtering by Stages.* Paper presented at the Convergence Information Technology, 2007. International Conference on.

Mir, A. F., & Banday, T. M. (2010). Control of spam: a comparative approach with special reference to India. *Information & Communications Technology Law, 19*(1), 27-59. doi:10.1080/13600831003589350

Mojdeh, M., & Cormack, G. V. (2010). *Semi-supervised spam filtering using aggressive consistency learning*. Paper presented at the Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, Geneva, Switzerland.

Moon, J., Shon, T., Seo, J., Kim, J., & Seo, J. (2004). An Approach for Spam E-mail Detection with Support Vector Machine and n-Gram Indexing. In *Computer and Information Sciences - ISCIS 2004* (pp. 351-362).

Mussa, D., & M. Jameel, N. (2019). Relevant SMS Spam Feature Selection Using Wrapper Approach and XGBoost Algorithm. *Kurdistan Journal of Applied Research, 4*, 110-120. doi:10.24017/science.2019.2.11

Nagamalai, D., Dhinakaran, C., & Lee, J. K. (2007). *Multi Layer Approach to Defend DDoS Attacks Caused by Spam.* Paper presented at the Multimedia and Ubiquitous Engineering, 2007. MUE '07. International Conference on.

Nakulas, A., Ekonomou, L., Kourtesi, S., Fotis, G., & Zoulias, E. (2009). A Review of Techniques to Counter Spam and Spit. In N. Mastorakis, V. Mladenov, & V. T. Kontargyri (Eds.), *Proceedings of the European Computing Conference* (Vol. 27, pp. 501-510): Springer US.

Namestnikova, M. (2012). Spam in July 2012. *Analysis,* (July 2012). Retrieved from http://www.securelist.com/en/analysis/204792243/Spam_in_July_2012

Navlani, A. (2018). KNN Classification using Scikit-learn. *Tutorials.* Retrieved from https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Nazirova, S. (2011). Survey on spam filtering techniques. *Communications and Network, 3*, 153+. Retrieved from http://go.galegroup.com/ps/i.do?id=GALE%7CA281789914&v=2.1&u=uwsydney&it=r&p=AONE&sw=w

Nelson, B., Barreno, M., Jack Chi, F., Joseph, A. D., Rubinstein, B. I. P., Saini, U., . . . Xia, K. (2009). Misleading Learners: Co-opting Your Spam Filter. In *Machine Learning in Cyber Trust* (pp. 17-51).

Nhung, N., & Phuong, T. (2007). An Efficient Method for Filtering Image-Based Spam E-mail. In *Computer Analysis of Images and Patterns* (pp. 945-953).

OECD, & Ahn, S.-i. (2004). Background Paper For The OECD Workshop On Spam [unclassfied]. doi:DSTI/ICCP(2003)10/FINAL

OstermanResearch. (2011). Spam Morphs from a Nuisance to a Threat [WHITE Paper]. *Osterman Research Security White papers,* (Dec 2011), 12. Retrieved from http://www.ostermanresearch.com/whitepapers/orwp_0153.pdf

Paswan, M. K., Bala, P. S., & Aghila, G. (2012, 30-31 March 2012). *Spam filtering: Comparative analysis of filtering techniques.* Paper presented at the Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on.

Patidar, V., Singh, D., & Singh, A. (2013). A Novel Technique of Email Classification for Spam Detection. *International Journal of Applied Information Systems, 5*, 15-19. doi:10.5120/ijais13-450976

Pelletier, L., Almhana, J., & Choulakian, V. (2004). *Adaptive filtering of spam.* Paper presented at the Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on.

Pennington, J., Socher, R., & Manning, C. (2014). *Glove: Global Vectors for Word Representation* (Vol. 14).

Pham, X., Lee, N.-H., Jung, J., & Sadeghi-Niaraki, A. (2011). Collaborative spam filtering based on incremental ontology learning. *Telecommunication Systems*, 1-8. doi:10.1007/s11235-011-9513-5

Pitsillidis, A., Kanich, C., Voelker, G. M., Levchenko, K., & Savage, S. (2012). *Taster's choice: a comparative analysis of spam feeds*. Paper presented at the Proceedings of the 2012 ACM conference on Internet measurement conference, Boston, Massachusetts, USA.

Pour, A., & Kholghi, R. (2012). Minimizing the Time of Spam Mail Detection by Relocating Filtering System to the Sender Mail Server. *arXiv.org, 4*(2). doi:10.5121/ijnsa.2012.4204

Qiu, X., Hao, J., & Chen, M. (2004). *Flow-based anti-spam.* Paper presented at the IP Operations and Management, 2004. Proceedings IEEE Workshop on.

Quinten, V., van de Meent, R., & Pras, A. (2007). Analysis of Techniques for Protection Against Spam over Internet Telephony. In *Dependable and Adaptable Networks and Services* (pp. 70-77).

Radicati. (2017). *Email Statistics Report, 2017-2021*. Retrieved from http://www.radicati.com/wp/wp-content/uploads/2017/01/Email-Statistics-Report-2017-2021-Executive-Summary.pdf

Radicati. (2018). *Email Statistics Report, 2018-2022*. Retrieved from https://www.radicati.com/wp/wp-content/uploads/2018/01/Email_Statistics_Report,_2018-2022_Executive_Summary.pdf

Radicati. (2019). *Email Statistics Report, 2019-2023*. Retrieved from https://www.radicati.com/wp/wp-content/uploads/2018/12/Email-Statistics-Report-2019-2023-Executive-Summary.pdf

Rajendran, B., & Pandey, A. K. (2012). Contextual Strategies for Detecting Spam in Academic Portals

Advances in Computer Science and Information Technology. Computer Science and Engineering. In N. Meghanathan, N. Chaki, & D. Nagamalai (Eds.), (Vol. 85, pp. 250-256): Springer Berlin Heidelberg.

Rao, J. M., & Reiley, D. H. (2012). The Economics of Spam. *Journal of Economic Perspectives, 26*(3), 87-110. doi:10.1257/jep.26.3.87

Rayana, S., & Akoglu, L. (2016). Less is More: Building Selective Anomaly Ensembles. *ACM Trans. Knowl. Discov. Data, 10*(4), 1-33. doi:10.1145/2890508

Ridzuan, F., Potdar, V., & Talevski, A. (2010). *Factors Involved in Estimating Cost of Email Spam*, Berlin, Heidelberg.

Robinson, G. (2003). A Statistical Approach to the Spam Problem. Retrieved from http://www.linuxjournal.com/article/6467

saeedian, M. F., & Beigy, H. (2009, March 30 2009-April 2 2009). *Dynamic classifier selection using clustering for spam detection.* Paper presented at the 2009 IEEE Symposium on Computational Intelligence and Data Mining.

Sanz, E. P., Gómez Hidalgo, J. M., & Cortizo Pérez, J. C. (2008). Chapter 3 Email Spam Filtering. In *Advances in Computers* (Vol. 74, pp. 45-114): Elsevier.

Schryen, G. (2007). *Anti-spam measures: Analysis and design*.

Shajideen, N. M., & Bindu, V. (2018). Conventional and Ontology Based Spam Filtering. In (pp. 1-3).

Sharma, A. K., & Yadav, R. (2015, 4-6 April 2015). *Spam Mails Filtering Using Different Classifiers with Feature Selection and Reduction Technique.* Paper presented at the 2015 Fifth International Conference on Communication Systems and Network Technologies.

Sharma, M., & Kaur, J. (2015). *A Novel Data Mining Approach for Detecting Spam Emails using Robust Chi-Square Features*. Paper presented at the Proceedings of the Third International Symposium on Women in Computing and Informatics, Kochi, India.

Sheu, J.-J., Chu, K.-T., Li, N.-F., & Lee, C.-C. (2017). *An efficient incremental learning mechanism for tracking concept drift in spam filtering* (Vol. 12).

Singh, A., & Batra, S. (2018). Ensemble based spam detection in social IoT using probabilistic data structures. *Future Generation Computer Systems, 81*, 359-371. doi:https://doi.org/10.1016/j.future.2017.09.072

Siponen, M., & Stucke, C. (2006). *Effective Anti Spam Strategies in Companies: An International Study.* Paper presented at the 39th Hawaiia International Conference on System Sciences.

Sirisanyalak, B., & Somit, O. (2007). *An artificial immunity-based spam detection system.* Paper presented at the Evolutionary Computation, 2007. CEC 2007. IEEE Congress on.

Smith, R. (2016). 40 years on from the first spam email, what have we learned? Here are 5 things you should know about junk mail. *Formative Content*. Retrieved from World Economic forum

So Young, P., & Shin Gak, K. (2008, 17-20 Feb. 2008). *Labeling System for Countering SIP spam.* Paper presented at the Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on.

Sophos. (2013). *Security Threat Report*. Retrieved from

SpiderLabs. (2013). Spam Types. Retrieved from https://www.trustwave.com/support/labs/spam_types.asp

Stern, H. (2008). *A Survey of Modern Spam Tools*.

Suryawanshi, S., Goswami, A., & Patil, P. (2019, 13-14 Dec. 2019). *Email Spam Detection : An Empirical Comparative Study of Different ML and Ensemble Classifiers.* Paper presented at the 2019 IEEE 9th International Conference on Advanced Computing (IACC).

Symantec. (2016). *Internet Security Threat Report*. Retrieved from

Takemura, T., & Ebara, H. (2008). *Spam Mail Reduces Economic Effects*. Paper presented at the Second International Conference on the Digital Society, Sainte Luce, Martinique

Takesue, M. (2010, 18-25 July 2010). *Cascaded Simple Filters for Accurate and Lightweight Email-Spam Detection.* Paper presented at the 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies.

Takumi, I., Akira, H., Yoshiaki, K., Ichimura, T. A. I. T., Hara, A. A. H. A., & Kurosawa, Y. A. K. Y. (2007). *A classification method for spam e-mail by Self-Organizing Map and automatically defined groups.* Paper presented at the Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on.

Tatyana Shcherbakova, Maria Vergelis, & Demidova, N. (2015). Spam and phishing in Q2 2015. *Quaterly Spam Reports,* (Q2, 2015). Retrieved from https://securelist.com/files/2015/08/KL_Q2_2015_SPAM_REPORT_ENG.pdf
https://securelist.com/analysis/quarterly-spam-reports/71759/spam-and-phishing-in-q2-of-2015/

Thornton, C. J. (1992). *Techniques in computational learning : an introduction*. London: London : Chapman & Hall Computing.

Tretyakov, K. (2004). Machine Learning Techniques in Spam Filtering. *InData Mining Problem-oriented Seminar MTAT, 3*.

Trivedi, S. K., & Dey, S. (2013, 3-5 Dec. 2013). *An Enhanced Genetic Programming Approach for Detecting Unsolicited Emails.* Paper presented at the 2013 IEEE 16th International Conference on Computational Science and Engineering.

Trivedi, S. K., & Dey, S. (2014). *A study of ensemble based evolutionary classifiers for detecting unsolicited emails*. Paper presented at the Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, Towson, Maryland.

V. Metsis, I. A. a. G. P. (2006). *Spam Filtering with Naive Bayes - Which Naive Bayes?* Paper presented at the 3rd Conference on Email and Anti-Spam (CEAS 2006), Mountain View, CA, USA.

Vapnik, V. (1998). The Support Vector Method of Function Estimation. In J. A. K. Suykens & J. Vandewalle (Eds.), *Nonlinear Modeling: Advanced Black-Box Techniques* (pp. 55-85). Boston, MA: Springer US.

Varghese, R., & Dhanya, K. A. (2017, 5-7 Jan. 2017). *Efficient Feature Set for Spam Email Filtering.* Paper presented at the 2017 IEEE 7th International Advance Computing Conference (IACC).

Volz, B., Behrendt, K., Mielenz, H., Gilitschenski, I., Siegwart, R., & Nieto, J. (2016). *A data-driven approach for pedestrian intention estimation*.

Vu Duc, L., & Truong Nguyen, V. (2012, 12-14 June 2012). *Bayesian spam filtering for Vietnamese emails.* Paper presented at the Computer & Information Science (ICCIS), 2012 International Conference on.

Wang, P., Xu, J., Xu, B., Liu, C., Zhang, H., Wang, F., & Hao, H. (2015). *Semantic Clustering and Convolutional Neural Network for Short Text Categorization*.

Wang, W. (2010, 18-23 July 2010). *Heterogeneous Bayesian ensembles for classifying spam emails.* Paper presented at the The 2010 International Joint Conference on Neural Networks (IJCNN).

Wei, Z., Feng, G., Di, L., & Feng, X. (2010, 7-9 July 2010). *Active learning based spam filtering method.* Paper presented at the Intelligent Control and Automation (WCICA), 2010 8th World Congress on.

Wijaya, A., & Bisri, A. (2016). *Hybrid Decision Tree and Logistic Regression Classifier for Email Spam Detection*.

Wijaya, A., & Bisri, A. (2016). *Hybrid Decision Tree and Logistic Regression Classifier for Email Spam Detection*.

Wu, C.-H., & Tsai, C.-H. (2008). Robust classification for spam filtering by back-propagation neural networks using behavior-based features. *Applied Intelligence*. Retrieved from http://dx.doi.org/10.1007/s10489-008-0116-0

Xiao, L., Junyong, L., & Meijuan, Y. (2010, 22-23 May 2010). *E-Mail Filtering Based on Analysis of Structural Features and Text Classification.* Paper presented at the Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on.

Xiao-wei, W., & Zhong-feng, W. (2012, 3-5 March 2012). *Good word attack spam filtering model based on artificial immune system.* Paper presented at the Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on.

Yang, L., Bin-Xing, F., & Li, G. (2006). *TTSF: A Novel Two-Tier Spam Filter.* Paper presented at the Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT '06. Seventh International Conference on.

Yang, Z., Nie, X., Xu, W., & Guo, J. (2006, 16-18 Oct. 2006). *An Approach to Spam Detection by Naive Bayes Ensemble Based on Decision Induction.* Paper presented at the Sixth International Conference on Intelligent Systems Design and Applications.

Yiu, T. (2019). Understanding Random Forest-How the Algorithm Works and Why it Is So Effective. *Towards Data Sceince*. Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Youn, S., & McLeod, D. (2007). A Comparative Study for Email Classification. In K. Elleithy (Ed.), *Advances and Innovations in Systems, Computing Sciences and Software Engineering* (pp. 387-391): Springer Netherlands.

Yu, B., & Xu, Z.-b. (2008). A comparative study for content-based dynamic spam classification using four machine learning algorithms. *Knowledge-Based Systems, 21*(4), 355-362. doi:10.1016/j.knosys.2008.01.001

Yue, X., Abraham, A., Chi, Z.-X., Hao, Y.-Y., & Mo, H. (2007). Artificial immune system inspired behavior-based anti-spam filter. *Soft Computing - A Fusion of Foundations, Methodologies and Applications, 11*(8), 729-740. doi:10.1007/s00500-006-0116-0

Zhang, Y., Liu, P., & Yao, J. (2019). Three-way Email Spam Filtering with Game-theoretic Rough Sets. In (pp. 552-556).

Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.

Zhang, Y., & Wallace, B. (2016). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *arXiv.org*.

Zhang, Y., Yang, X., & Liu, Y. (2012, 29-31 Dec. 2012). *Improvement and optimization of spam text filtering system.* Paper presented at the Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on.

Zheleva, E., Kolcz, A., & Getoor, L. (2008). Trusting spam reporters: A reporter-based reputation system for email filtering. *ACM Trans. Inf. Syst., 27*(1), 1-27. doi:10.1145/1416950.1416953

Zhen, Y., Xiangfei, N., Weiran, X., & Jun, G. (2006, 3-6 Nov. 2006). *Application of the Character-Level Statistical Method in Text Categorization.* Paper presented at the Computational Intelligence and Security, 2006 International Conference on.

Zhijun, L., Weili, L., Na, L., & Lee, D. A. L. D. (2005). *Detecting and filtering instant messaging spam - a global and personalized approach.* Paper presented at the Secure Network Protocols, 2005. (NPSec). 1st IEEE ICNP Workshop on.

Zhou, B., Yao, Y., & Luo, J. (2014). Cost-sensitive three-way email spam filtering. *J Intell Inf Syst, 42*(1), 19-45. doi:10.1007/s10844-013-0254-7

Zhou, Y., Jorgensen, Z., & Inge, M. (2007, 29-31 Oct. 2007). *Combating Good Word Attacks on Statistical Spam Filters with Multiple Instance Learning.* Paper presented at the 19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007).

Zhou, Y., Jorgensen, Z., & Inge, M. (2008). Countering Good Word Attacks on Statistical Spam Filters with Instance Differentiation and Multiple Instance Learning. In.

Zhu, F. (2018). A Classification Algorithm of CART Decision Tree based on MapReduce Attribute Weights. *International Journal of Performability Engineering, 14*. doi:10.23940/ijpe.18.01.p3.1725