Department of Computing and Information Technology

Unitec Institute of Technology

Auckland, New Zealand

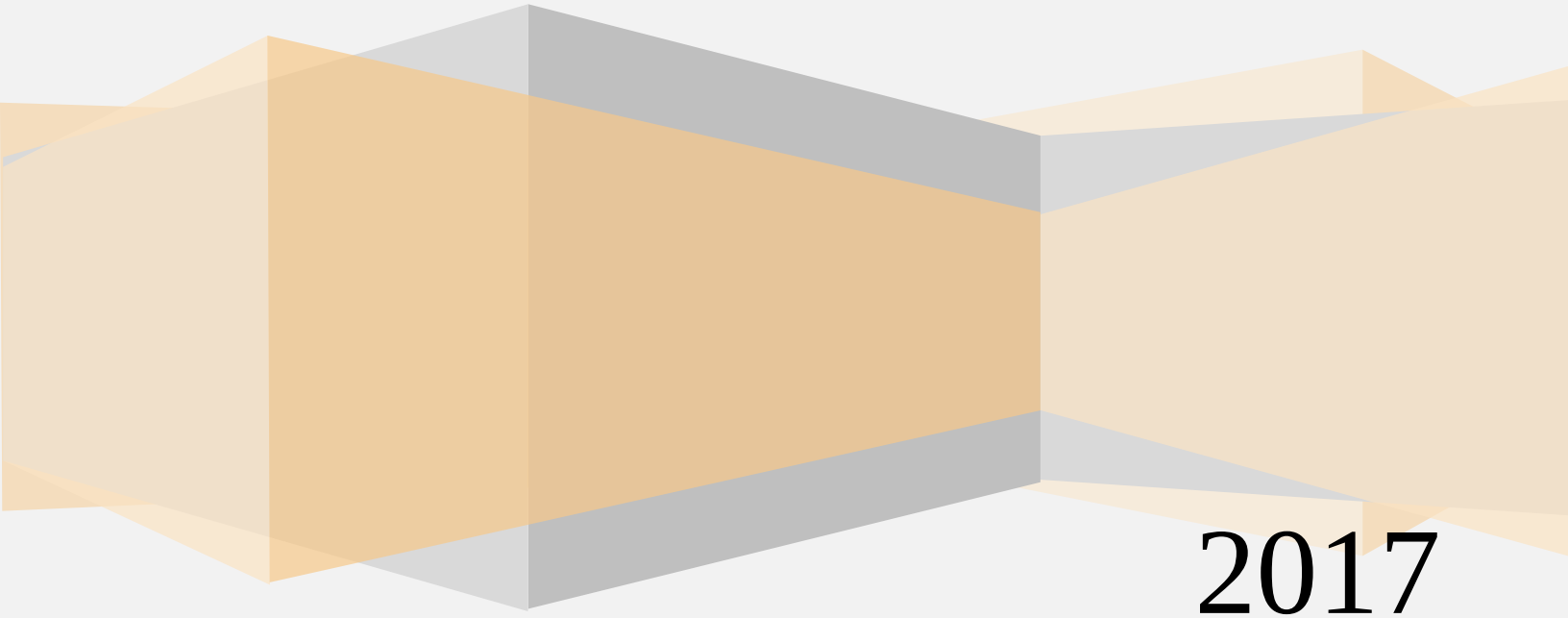
Evaluating Spammer detection systems for Twitter

A thesis submitted for the degree of

Master of Computer

Student Name: Trung Minh Ho

Student ID: 1396701



2017

Abstract

Twitter is a popular Social Network Service. It is a web application with dual roles of online social network and microblogging. Users use Twitter to find new friends, update their activities or communicate with each other by posting tweets. Its popularity attracts many spammers wanting to spread advertising or malware. Many systems are proposed to detect spammers or spam tweets using a different subset of features, or by extracting the features based on different numbers of recent tweets. However, we do not know which proposed system is the best, because they use different techniques, such as different subsets of features, number of recent tweets, classifiers and evaluation metrics. Over time, spammers will change their key features to disguise themselves as a normal user, and we do not know whether the current systems are able to cope well with this phenomenon, which is called spam drift.

In this research we have created a tool called WEST, which stands for "Workbench Evaluation Spammer detection system in Twitter." This tool allows users to investigate their model's performance against the spam drift problem and save much time for further users to do their research in this field; for example, extracting the features, which is a time-consuming process. Also, we did comprehensive investigative studies on the proposed 172 features, which include content-based and user-based features from the existing systems to find the optimised subset of features that is effective, efficient and resilient at detecting spammers.

Based on the investigation of the 172 existing features, we found a model that we called ASDF, which stands for Anti Spam-Drift Features, that could detect spammers at 91% True Positive rate and performed the best at handling the spam drift problem compared to the existing spammer detection systems.

Keywords: Spammer detection, spam drift, Twitter, optimisation subset of features, optimisation recent tweets

Acknowledgements

I would never have been able to finish my dissertation without the guidance of my supervisors and support from my family.

I take this opportunity to express my sincerest gratitude to my principal supervisor, Dr Veronica Liesaputra, for her professional support, treasured comments, and, at many stages of this research project, I benefitted from her advice, particularly when exploring new ideas. I have been extreme lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly.

Furthermore, I would like to express thanks to my associate supervisors, Dr Mahsa Mohaghegh and Dr Sira Yongchareon, for their advice, dedication, valuable comments, guidance and for going through the thesis and providing very helpful and encouraging feedback. Again, thank you so much to my supervisors for being there for me.

Lastly, I would also like to thank my parents. They were always supporting me and encouraging me with their best wishes.

Table of Contents

List of Tables

List of Figures

List of Equations

List of Abbreviations

IBK	K-nearest neighbors
J48	Decision Tree
NB	Naive Bayes
RF	Random Forest
SVM	Support Vector Machine
RT	Recent Tweets
ASDF	Anti Spam-Drift Features
WEST	Workbench Evaluation Spammer detection system in Twitter
ARFF	Attribute-relation file format
WEKA	Waikato Environment for Knowledge Analysis

CHAPTER 1

INTRODUCTION

Twitter is a popular Social Network Service that allows users to post messages, called tweets, of up to 140 characters. It allows the users to share news, opinions, trending topics, photos, or general content [1]. Spammers exploit Twitter functions to spread malicious content. Twitter reported that in 2014 around 13.5 million accounts were either spam or fake [2], and according to [3] a study shows 90% of users click on a new spam link before this link is blocked by the blacklist. There are many proposed systems for detecting spammers in Twitter. However, spammers continue to evade existing techniques by changing their behaviour or changing their key features to disguise as a legitimate user [4], and this phenomenon is called spam drift. Furthermore, it is hard to find the best systems to identify spammers because they use different evaluation setups. Thus, it is our goal to investigate what are the best features to identify spammers as none of the researchers are doing any investigation about it. In order to achieve this goal, we have created a tool called WEST (Workbench Evaluation Spammer detection system in Twitter). This tool contains current techniques to create a spammer detection system in Twitter. Moreover, this is a novel tool for researchers to evaluate their proposed system to determine whether their system performance is good against the spam drift problem or not.

1.1 Background of the study

Twitter is a web application with dual roles of online social networking and microblogging [5]. Twitter users use the platform to find new friends, update their activities [6] or communicate with each other by posting tweets [5]. When a user posts a tweet, it immediately appears to their followers or followings and allows them to spread the received information [7].

Twitter has become one of the most popular Social Network Services [8]. Its popularity attracts many spammers wanting to spread advertising or malware. About 83% of the users of social networks have received at least one unwanted friend request or message [9], and 45% of users on social networks click on links posted by friends in their friend list accounts, even though they do not know those people in real life [6]. Users identify spammers manually based on their experience, which can lead to problems with false positives. Therefore, it is important to have a tool that can automatically identify spammers [6]. These goals are similar to traditional spam emails, but spam in Twitter is different because Twitter limits each message to only 140 characters. Therefore, spammers in Twitter cannot put lengthy information in each tweet [10], while emails consist of headers, subject and body, which alone contain more content than a tweet.

A number of researchers extracted features from tweets that were gathered from Twitter API and applied machine learning algorithms for spammer detection. [7] used the maximum number of tweets, age of account, or number of followers to detect spammers by using SVM. [11] applied several machine learning algorithms such as SVM, NB, DT and RF to detect spammers based on

profile description, or number of tweets per day. However, the statistical attributes, such as number of words, number of characters, or number of tweets posted in the early morning vary over time, for example, spammers always post more tweets in the early morning, and now they have changed it by posting tweets at a random time of day, and because of that, the performance decreases due to new tweets come for identification [12]. Also, spammers will try to find a way to lure the current systems by disguising the key features, and then the detection method will fail soon [13]; this phenomenon is called "Spam Drift".

1.2 Problem Statement

After studying the existing methods, we found that many spammer detection systems utilise different features to detect spammers and most were expanded from #hashtag, @mention and URL. Also, those features were extracted from different numbers of recent tweets, such as in [14] where the authors counted *the number of #hashtags* from all the tweets for one user, [6] counted *the number of #hashtags* from 100 recent tweets, and [15] counted *the number of #hashtags per word.* All seems to perform well, but we do not know which one is the best and why, or how well they handle spam drift. Therefore, in this research, we present a problem statement:

"What is the most effective, efficient and resilient system for detecting spammers in Twitter?"

An effective and efficient spammer detection system in Twitter must be able to identify spammers with high overall true positive rate, recall, precision, f-measure and accuracy in the least amount of time possible. One factor that could influence the effectiveness of a system is the subset of features selected for training and building the model. Also, the performance of the current features is dramatically influenced by spam drift as mentioned. Statistical features used to detect spam vary over time as spammers continuously change strategies. They try to evade a spammer detection system by making those statistical attributes resemble normal tweets. Thus, a resilient spammer detection system must be a system that can handle spam drift.

The term "efficient" relates to the time to build and classify a system, and the factors that affect the "efficiency" are the number of recent tweets for feature extraction and classifiers for training. Finally, a "resilient" system must be able to detect spammers with a high true positive rate after a certain period of time, even though spammers will change their behaviour or key features to disguise themselves as a legitimate user. Thus, in this research, we would like to find the optimisation subset of features and techniques to identify spammers. In the next section, we present three research questions that are used to find the answer to the problem statement.

1.3 Research Questions

Apart from creating the WEST tool and publishing conference papers, another objective of this research is to investigate the best features for identifying spammers in Twitter. While looking at the existing methods in this field, we found that the most important aspects for building a spammer detection system are the feature extraction and feature selection steps. Because the existing methods experimented with different datasets and there is no evaluation workbench that enables

us to compare each model, it is hard to objectively determine all the contributing factors for producing the most effective and efficient spammer detection model. Thus, we formulated our research questions based on this motivation.

Question 1: What are the most effective content-based and user-based features for detecting spammers in Twitter?

There are many content-based and user-based features that are used to build a spammer detection system in Twitter, but we do not know which are the best subsets of features for building a system. The term "effective" means the proposed model must detect spammers with high Accuracy, True Positive Rate, F-measure, Recall and Precision, regardless of the time required to extract and classify them. In order to answer this question, we will collect many different features from many existing systems and perform several experiments to find the optimal subset of features. We have reviewed 172 content-based and user-based features, from many authors, concerning how to extract them. In Chapter 3, we will explain how to perform the experiments to find the optimal subset of features and how to extract the 172 content-based and user-based features.

Question 2: Which model is the most efficient at identifying spammers in Twitter?

Although many researchers have proposed various systems for detecting spammers in Twitter, none have mentioned the time it requires to achieve this. It is important to know the time because in November 2016, on average, there were around 6,000 tweets every second [16], so it is essential for legitimate users to quickly determine if tweets are sent by a spammer.

The most efficient spammer detection system should be able to detect spammers accurately in the least time possible. The factors that might affect the system's efficiencies are feature selection, the number of recent tweets and classifiers. For feature selection, we measure how long it takes to extract the features and how useful the features are at detecting spammers. With each classifier, we measure the time required to build a model and generate the results based on the feature selected, and how well the model performed.

The number of recent tweets is the number of tweets used to extract the features, for example, [6] used 100 recent tweets to extract the features, and [17] used 20 recent tweets for their system. We hypothesize that the larger the number of recent tweets a system uses, the longer it will take to extract the features. To answer this question and find out the best technique that requires the least amount of time, we will apply some experiments to different groups of features for different numbers of recent tweets.

Question 3: Which model is the most resilient at handling the spam drift phenomena?

The performance of a spammer detection system will decrease over time because spammers always change the properties of the new incoming tweets to avoid being detected, i.e. those systems are not resilient [6, 17]. It is important for people to be able to quickly evaluate the effectiveness of their model at handling those tweets and make changes to their model when it is not effective anymore. However, currently, there is no easy way for people to do so. To be resilient, they have

to regularly update their training datasets and implement new features that they should use in their current model. Usually, those features are the ones that have been proposed by other researchers.

Chapter 3 shows how we can use our novel tool, called WEST, to not only help researchers investigate how well their model is at handling tweets from various time periods, but also to enable them to leverage the findings of other researchers and try out features proposed by other researchers that can better handle the spam drift problem. With WEST, people can easily compare the performance of two or more spammer detection systems and deduce which model is the most resilient. Section 4.3 illustrates the results of our spam drift evaluation.

1.4 Research Contribution

This research presents a comprehensive study of the existing systems used to identify spammers, irrespective of the dates the tweets were created in Twitter and finding a resilient model that can capture spammers with the highest accuracy in the least amount of time possible. This model will be created based on the collection of features that have been used by the existing spammer detection systems. The proposed model in this research will contribute as an alternative approach in this field, and also expand the knowledge for further research.

In addition to finding a new model to identify spammers, a workbench tool is developed which supports the current techniques such as 172 built-in content and user-based features, different number of recent tweets, and feature selection algorithms and classifiers. This tool allows the users to perform the experiments easily and also will contribute to the public as open-source.

We published two conference papers: *Evaluating Social Spammer Detection Systems* [18] presents the most effective and efficient features to build a spammer detection system in Twitter, and *A Framework for Evaluating Anti Spammer Systems for Twitter* [19] evaluates the impact of using different numbers of recent tweets to obtain a faster and more accurate model. These papers provide knowledge for future researchers to find the most effective model to create a spammer detection system for Twitter. [18] was published in a CORE A rank conference.

1.5 Thesis Structure

The thesis is organised into five chapters as outlined below:

Chapter 1: An introduction to spammers in Twitter and the research objectives. This chapter outlines the research questions in order to answer the problem statement of finding the most effective, efficient and resilient model to identify spammers in Twitter.

Chapter 2: Reviews of spammers on different social media platforms, such as spam in email, YouTube and Twitter, and the methods used to identify spammers in those social media. Also, this chapter explains in detail the existing techniques to create a spammer detection model, such as feature extraction, feature selection algorithms, and evaluation metrics to evaluate the performance of a system.

Chapter 3: Presents the methodology to find the most effective, efficient and resilient model to detect spammers. These methods include how to extract the features from a provided dataset, and are followed by experiments to find the optimisation subset of features, the number of recent tweets, and classifiers. Lastly, a comparison between the proposed model and the existing systems is presented.

Chapter 4: Discusses the experimental results and explains in detail for every experiment and shows the ASDF model against spam drift in Twitter.

Chapter 5: The conclusion of the thesis and discussion of improvements are mentioned in this chapter.

CHAPTER 2

LITERATURE REVIEW

The objective of this research is to find an efficient, effective and resilient system for identifying spammers in Twitter. To answer this objective, we have formulated three research questions to find out the best techniques to build that kind of system based on existing techniques. This chapter presents the existing methods related to techniques for detecting spammers on Social Network Services (SNS), such as Twitter, YouTube and Email, and we aim to understand the existing techniques that have been used to build a spammer detection system. We will use those existing techniques to create the WEST that enables us to find a resilient model for detecting spammers and answering the research questions mentioned in Section 1.3.

The first research question investigates the collected features to find a good subset of features to identify spammers. In this chapter, we have collected 172 content-based and user-based features, and we will use our novel tool, WEST, to apply some feature selection algorithms to find the most relevant features related to the model. Some of the features might not be relevant to the model; therefore, it is important to remove the irrelevant features to improve the performance of the system.

The second research question finds the most efficient model to identify spammers. An efficient model must detect spammers with high levels of accuracy and in the least time possible. This depends on feature selection, the number of recent tweets and classifiers. This chapter shows some of the feature selection algorithms used to pick the optimisation subset of features, several number of recent tweets for feature extraction used by the existing spammer detection systems, and the classifiers used for classification. We will use WEST to perform the experiments to find the best model to detect spammers.

The last research question finds a resilient system that could handle the spam drift problem. A resilient system must be able to detect spammers with high accuracy when a dataset from a different time period is loaded. With WEST, we can perform experiments to see what features are effective at overcoming the spam drift problem.

Chapter 2 is structured as follows. Section 2.1 shows the existing works related to detecting spammers in Twitter. Section 2.2 introduces spam in other social media and the existing methods used to identify spammers in those media. Section 2.3 demonstrates how to collect a dataset, the format of the dataset, and how to classify it into spam and non-spam. Section 2.4 explains the 172 content-based and user-based features collected for this research and some features that we proposed. Section 2.5 describes the feature selection algorithms used to find the relevant features for a model. Section 2.6 shows the common evaluation metrics to evaluate the performance of a system. Section 2.7 mentions the classifiers for training and classifying to find spammers in Twitter and lastly, Section 2.8 summarises this chapter.

2.1 Spam in Twitter

Users spend a significant amount of time on SNS. Twitter is one of the famous SNS, and millions of the users login daily who encounter spam. They use Twitter for storing and sharing personal information with their friends, or people around the world. This attracts the interest of cybercriminals, such as spammers, who find the personal information valuable for identity theft [9]. Spam is an unwanted activity, such as when marketers send members unwanted advertisements or steal user information by directing users to malicious external pages [20].

In SNS, spammers employ many techniques to post unwanted messages to users on SNS, such as Twitter, for advertisements, scams or spreading of malware through malicious URLs [21]. It is easy for humans to distinguish spammers and legitimate users, but it wastes user time and puts them at risk of accessing malicious content [1]. Hence, spam is becoming a significant problem for users, and therefore many approaches have been proposed to determine whether a user is a spammer or not [8]. In our research, detecting a spammer is more important than detecting a spam tweet, because spammers tend to send multiple spam tweets.

[22] show that Machine Learning provides powerful techniques for spammer detection in Twitter. A spam detection system could use user-based features (which are extracted from user's behaviour) or content-based (which are extracted from linguistic features of a tweet) [23]. For example, [1] find spammers' profiles by extracting features from user profiles and training the features with a supervised machine learning technique, such as NB and SVM. They evaluate the performance of their techniques based on precision (percentage of positive prediction that is correct), recall (percentage of positive instance that is predicted as positive), and accuracy (percentage of prediction that is correct). [6] extracted the user-based and content-based features from the dataset, then ran it with a RF classifier and evaluated the result using the precision and F-measure (the harmonic mean of precision and recall).

Spam Drift

While researchers are continuously working to improve the accuracy of spammer detection systems, spammers are also tirelessly trying to avoid being detected. Spammers would regularly change the behaviour or characteristics of their tweets, for example, posting more tweets or creating spam tweets with the same meaning but in different words. Thus, the statistical features useful for detecting spammers and spam tweets are changing over time, and this is known as "Spam Drift" [24].

In the example below, it can be seen that tweets1 and tweet2 are having a similar content and these are the spam tweets. However, the difference between these two tweets is that tweet1 has an URL, while tweet2 did not contain any URL. In fact of a system is using an URL-relation feature to identify spammers, for example, counting *Number of URL per tweet* because [25] said that spammers are normally using URL in their tweet to increase the change of getting clicked by the normal users, therefore in this case, the system will ignored tweet2 as a spam tweet because it is not containing any URL in the tweet. This is an example of the problem in spam drift phenomena.

For example:

tweet1: *Just finished a call. Iâ€™m available now on #NiteFlirt 1-800-To-Flirt, ext: 01617535. Give me a call before... http://t.co/HB9gZrs5bl*.

tweet2: *Just finished a call. I'm available now on #NiteFlirt 1-800-To-Flirt, ext: 01617535. Give me a call before I become busy again!*

To tackle this problem, [24] update their system frequently with new training data. First, they labelled their training data and extracted the features, then applied machine learning to create a spammer detection system. Then, updated the training set by adding high-purity spam tweets and training the system again every three days with the same features. They have collected their dataset in a period of 10 consecutive days. Then they used tweets posted in Day 1 for training purposes and Day 2 to Day 10 for testing. They found that the performance of their model is changing over time, for example, the detection date of Day 3 (almost 80%) is slightly better than Day 2 (about 76%), while Day 7 (78%) is about same as Day 6 (77%), and there is a big gap between Day 6 and Day 3. Also, they have reported that 92% spamming account will be suspended by Twitter within three days. Therefore, they believed that spammers are most likely to change spamming strategies. Because of that, they decided to update their training set in every 3 days because if they update their training set in a short time period for example, 1 or 2 days then the re-training process will be done more frequently, which is time-consuming. On the other side, if it is too long for updating the re-training process then the classifier will become less accurate [24].

[26] proposed three different models: (1) a time-window based model where the system regularly trains the system based only on the recent information given in the specified time-window, and disregarding the old information; (2) an incremental model where the system regularly trains the system based on all of the information they have to date, i.e. old information is retained and the latest information is added to the training dataset, which would lead to storage problems; and finally, (3) an ensemble-based model, which is the contribution to their work. The ensemble-based model can learn incrementally without storing previously seen data. The idea behind this model is that the use of a committee of classifiers can provide better results than the best of the single classifiers; it uses a voting algorithm to combine the output of multiple classifiers into a single decision. Their evaluation shows that the average time-window based model achieved 51.53%, the ensemble-based model achieved 60.31%, and the incremental-based model achieved 76.39%.

[3] introduced Lfun scheme to address the "spam drift" problem. This scheme contains two components: LDT (Learning from Detected Spam Tweets), that is, learning from detected spam tweets. LHL (Learning from Human Labelling) is learning from human labelling. There are three stages to this approach and in the first stage, they have labelled the training dataset into spam and ham profiles, then trained the LDT component with that dataset and updated the training set with the spam tweets detected by LDT component. In the second stage, they used the LHL component to differentiate unlabelled tweets into spam and non-spam profiles. For a small number of tweets, for example, 100 spam tweets, they asked humans to verify whether the tweets classified by LHL are correct or not. In the last stage, they combined the training dataset provided by LDT, and the small number of human-labelled tweets becomes the new training set for the RF classifier that will predict the incoming tweets as spam.

Although other researchers attempted to solve the spam drift issue by regularly retraining their model by regularly updating their training dataset, in this research, we hypothesise that if we had chosen the correct subset of features to detect spammers, we would be able to tackle the spam drift phenomenon without regularly rebuilding the model for classifying the spammers.

2.2 Spam in other media

Spam is not only a problem that occurs on SNS, but also in email and video sharing social networks such as YouTube and Email [27, 28]. In emails, spam can be used for spreading viruses, malicious code, or for fraud in banking [29]. Also, spammers may post unrelated video content with a popular title or name to increase views from users [30].

2.2.1 Spam Email Detection

Email is a popular form of communication, providing an easy and reliable method to communicate with many people, as one email can spread among millions of people in a moment [31]. Apart from the benefit that email provides to us, there is a problem for every email user [32], as shown in a study in March 2013, when about 100 billion spam emails were sent out, and many studies agree that spam emails have an attributable pattern [33]. To avoid spam emails, we need an effective spam filtering system [29]. However, there is no perfect solution to stop incoming spam emails [34] and many approaches have been proposed to classify spam emails.

A word-count algorithm could be used to extract the features from an email by removing *stop-words* (conjunctions, prepositions, and articles) and *non-words* (containing special symbols), then counting the total number of unique words out of the total words to find the frequency of that word. The main idea of this algorithm is to make a dictionary [32]. Next is training the data with a Naive Bayes classifier, which calculates the probability of spam or ham (non-spam) and if the probability of spam is greater than that of ham, it is considered as a spam email. To evade the system detection, spammers would intentionally misspell words, for example, coomputer or c@mputer instead of computer [35]. [28, 35] proposed using a word-stemming technique to replace consecutive repeated characters to a single character, for example, "discounts", "discounted", or "discounting" to "discount". More than that, [28] normalised all the dollar signs($) to be replaced with the text *\dollar*, then created a list of words for each email, then mapped each word in the preprocessed email with the vocabulary list (each word in the vocabulary list contains an index), and then trained the system with a Support Vector Machine classifier.

2.2.2 Spams in YouTube

YouTube is the most popular video-sharing social network. In May 2008, 74% of U.S social network users viewed 12 billion videos, and 34% of this number were from YouTube [36]. By allowing users to share their videos, users may become susceptible to different types of malicious content [30].

To find spammers on YouTube, [36, 37] proposed to extract a number of attributes from users (number of videos uploaded, and number of videos watched) and video features (number of views, and duration). However, this classifier is slow on a whole dataset with 264,460 users and not efficient on multiple classes for finding legitimate spammers and promoters. To solve this, [27] proposed the use of SVM-KNN which trains a Support Vector Machine on K-number of features of nearest neighbours. However, it appears that the users and videos feature is not useful in helping us find spammers, because Twitter does not have those attributes.

2.3 Dataset

There are many ways to collect the dataset, for example, using Twitter API [5, 14, 38] but it only allows 150 requests per hour. To overcome this limitation, four servers and 120 IP addresses can be deployed and make the servers change their IP address whenever the current IP address is reaching the limit [10]. Once a dataset is collected, it could be classified as spam or non-spam manually [6], or the Amazon Mechanical Turk (AMT) volunteers can be asked to evaluate every tweet and assign all tweets into desired groups [14]. We must pay to use AMT and it might take a long time for workers from AMT to evaluate thousands of tweets.

However, there are some public datasets available on the Internet that already classify into spam or non-spam; for example, [22] involved the dataset from [7] which includes 1065 users: 355 spammers and 710 non-spammers. In our system, we will use the dataset from [11] and [39] because these are the only datasets that we could find in a raw format and they are already classified into legitimate and spam users.

2.4 Feature Extraction

Feature extraction is a process of extracting features that can be utilised by a machine learning technique to produce accurate results [32]. In this section, commonly used features can be organised into two categories: content-based and user-based. User-based features are used for detecting the spammers, and content-based features are used for detecting the spam tweets. In this research, we do not use other based features, such as graph-based, because the dataset [11] and [39] that we will use for this research does not have enough information to extract the graph-based features, and also it takes a long time to download the required data; thus we only use content-based and user-based features.

2.4.1 Number of recent Tweets

Several authors have extracted features from different numbers of tweets recently. [25] used all tweets, [17] used the 20 most recent tweets and [6] used the 100 most recent tweets. However, these approaches used different techniques, so it is hard to determine the best numbers of recent tweets. However, this points out that we could use one of the above number of recent tweets for feature extraction, and we could also find the optimisation number of recent tweets in this research.

2.4.2 Content-Based features

Content-based features are linguistic features extracted from the content of tweets [15], and Table 1 shows a list of content-based features. Spammers often used a shortened URL service to lure legitimate users and Twitter does not check the legitimacy of shortened URLs. Additionally, spammers tend to use the same URLs in multiple tweets to increase the chance of getting it clicked by legitimate users [25]. Features that are related to URLs have been utilised by most researchers, for example, *Number of URLs* [38], *Number of URLs per word* [15], or *Number of unique URLs* [40].

The *#hashtags* is used to describe a term; if there are many tweets containing the same term, then it will become a trending topic [6]. Spammers often include a trending topic in their tweets with unrelated content to lure legitimate users to access their tweets [10]. Similar to *URL* features, *#hashtags* could be expanded to other forms such as *Number of #hashtags per word on each tweet* [15].

In Twitter, users can include *@username* in their tweets, called a *mention.* The mentions could be sent to any receivers even when they are not followers or followings of a sender. Spammers tend to use this function to send spam tweets [10]. This feature has been used by several authors, such as [6, 25].

Additionally, spammers often post a larger number of spam words in their tweets than normal users. [7] observed that 39% of spammers in their study posted all their tweets containing *spam words* but legitimate users do not post more than 4% of their tweets containing *spam words.* Therefore, a spam word feature is considered in some papers, for example, [6, 7] used a spam word list from Wordpress.org. One step further, [14] found the *percentage of words that are not found in a dictionary* and achieved only 80% accuracy.

In Table 1, "*The tweet (feature number 14)"* and "*Twitter client (feature number 30)"* features will not be extracted in this research because "*The tweet"* feature is not clearly explained and "*Twitter client*" feature is no longer supported by Twitter.

Table 1 - Table of content-based features

	Content-Based		
Feature No	**Feature name**	**Definition**	**References**
	URL/HTTP Links		
1	Number of URLs	Find out number of URLs on a tweet	[6, 14, 20, 38, 40, 41]
2	Number of URLs per word	Find out number of URLs per word on a tweet	[15]
3	Whether the link points to a Social media domain	Check if a URL is Twitter, Facebook, or YouTube...	[14]
4	Number of unique URLs	Number of unique URLs on a tweet	[40]
	Hashtags		
5	Number of hashtags per tweet	Find out number of #hashtags on a tweet	[6, 14, 15, 38, 41]
6	Number of hashtags per word on each tweet	Find out number of #hashtags per word on a tweet	[15]
	Mentions		
7	Number of mentions	Find out number of @mentions on a tweet	[6, 14, 15, 21, 40]
8	The number of mentions per word	Find out number of @mentions per word on each tweet	[7], [15]

	Special characters		
9	Number of exclamation marks	Find out number of exclamation marks on a tweet	[15]
10	Number of question marks	Find out number of question marks on a tweet	[15]
	Numeric		
11	Numeric characters	Find out number of numeric characters on a tweet, for example 1,2,3 etc.	[17]
	Retweets		
12	Whether the tweet is a retweet	Check if a tweet is a retweet	[14]
13	Number of Retweets	Find out a total number of retweets on a tweet	[6, 38]
	Tweets/Words		
14	The tweet	Determine the writing style.	[42]
15	Number of consecutive words	Count the total number of consecutive words on a tweet. Every two words is a consecutive word, for example, "Social Network Analysis" is two consecutive words.	[14]
16	Number of characters per tweet	Count the total number of characters in a tweet	[7, 38]
17	Number of whitespaces per tweet	Count the total number of blank spaces on a tweet	[15]
18	Number of capitalised words per tweet	Count the total number of Capitalised words on a tweet. If the first character of a word is a capital, then it is a capitalised word. For example, in "Social NETWORK aNalysis", the capitalised words are "Social" and "NETWORK".	[15]
19	Number of capitalised words per word on each tweet	Count the total number of Capitalised words per word on a tweet.	[15]
20	Duplicate Tweets	Find out the total number of duplicated tweets. If two tweets are exactly the same, then it is considered as a duplicated tweet.	[17]
21	Percentage of words not found in a dictionary	Find out the percentage of words not found in a dictionary on a tweet	[14]
22	Tweets contain Places	Check if a tweet contains any cities	[14]
23	Tweets contain Organisation	Check if a tweet contains any Organisation names	[14]
24	Tweets contain name	Check if a tweet contains any names for example,	[14]
25	Tweets contain social media domain	Check if a tweet contains any social media domain	[14]
26	Number of words	The total number of words on a tweet	[7], [15]
	Spam-words		
27	Number of spam words per tweet	Find out the total number of spam words on a tweet	[6, 7, 15]
28	Number of spam words per word on each tweet	Find out the total spam words per word on a tweet	[15]
29	Time of publication	The time a tweet has been posted	[42]
30	Twitter Client	Twitter client (i.e., Pcs, smartphone or laptop)	[42]

2.4.3 User-Based features

User-based features can be derived from specific properties of user behaviour [7]. Spammers try to follow a large number of users to gain their attention [6]. For user-based features, the number of followings, followers, and reputation are the common features that have been used together. The *following* feature means someone follows other accounts; the *follower* is the opposite of following, and the *reputation* determines a person influential on Twitter.

[6] proposed a Random Forest classifier approach to detect spammers, extracted *following* and *follower* features and achieved 95.7% precision in comparison. According to [43], this model used

followings, *followers* and *reputation* features crawled from Twitter API and achieved 100% precision based on Neural Network and SVM classifiers, however, in fact of choosing a consistent model, Naïve Bayes is better than Neural Network and SVM, because it is achieved 91.7% across the evaluation metrics. While [17] used the *followings*, *followers*, and *reputation* but achieved only 89% precision based on Naive Bayes classifier. [21] and [1] used the same *Followers-to-Following-Ratio* feature. Spammers attempt to follow many accounts and try to create a relationship with their followings, but it is hard to get the follow-back, and the feature makes sure that each account maintains a healthy ratio of their followers and followings to avoid being suspended, and [21] proposed an approach that can detect 93.6% of spammers while [1] achieved 89% precision.

On the other hand, [4] said that *the number of followings, followers and reputation* and *Followers-to-Following-Ratio* could be evaded since spammers could purchase *followers* from websites with a small amount of money. Thus, [4] introduced a new feature called *Bi-directional Links Ratio*; if two accounts are following each other, they are considered a Bi-directional link. The idea behind this feature is that even though spammers increase their number of Bi-directional links, they need to pay more to buy *followers* to evade this feature, so it is worthless for them [4]. Hence, it seems like the *Bi-directional Links Ratio* feature is effective, because it is hard for spammers to evade.

In many cases, spammers insert a malicious URL and convert it into a shortened form [40]. A study shows that spammers tend to post about 95% of tweets containing URLs, and normal users post only 7% of tweets containing URLs [44]. The following two efficient features, *URL rate* and *Interaction Rate*, have been proposed by [44]. The *Interaction Rate* is proved by [44] as an effective feature because normal users usually have interaction with their friends (followings or followers). In contrast, spammers do not have such interactions with normal users, and usually just post URLs and also spammers have a larger proportion of tweets containing URLs than normal users, so the *URL rate* is finding the number of tweets containing URLs over the total number of tweets posted by a user. It is hard to evade this feature, unless spammers just post spam tweets without any URL, so there is not much information in the tweets [44].

There is another effective feature called *Fraction Mention to Non-Follower*, which is expanded from the @mention feature. This feature represents a relationship between the sender and the receiver, and the connectivity and distance between a legitimate user and their followers is 1. However, spammers tend to use the *@mention* technique to random users, and the distance must be greater than 1 [10]. Table 2 shows all the user-based features we have collected during a review of the existing systems.

Table 2 - Table of user-based features

User-based features			
	Account profile features		
Feature No	Feature name	Definition	Reference
1	Number of followers	Find the number of followers of a user	[6, 7, 40, 41]
2	Number of followees	Find the number of followees of a user	[6, 7, 17, 40, 41]
3	Reputation	Find out the reputation of a user	[6, 7, 40]
4	Ratio follower to following	Find out the ratio of follower to followee of a user	[1, 7]
5	Age of account	Find out the age of an account	[7, 40]
6	Bi-directional Links Ratio	The number of followers that a user is following back	[4, 40]

7	Interaction rate	Check whether the user of a tweet reply is a friend of the observed account or not. If the user is in the friend list, then a tweet receives an interaction.	[44]
8	Fraction of mention non follower.	Find out the fraction of mentions sent to non follower.	[10]
	#HashTag relation features		
1	Average, Maximum, Minimum, Median number of hashtags	Find out the max, min, median, and average number of #hashtag used by a user	[44], 5]
2	Maximum, Minimum, Median number of hashtags per word	Find out the max, min, median number of hashtags per word used by a user	[7]
3	Fraction of hashtags	Find out the fraction of tweets with #hashtag on a user	[10]
4	Hashtag ratio	The ratio of tweets containing Hashtag	[10]
	@Mention relation features		
1	Average, Maximum, Minimum, Median number of mentions	Find out the average, max, min, and median number of @mentions of a user	[7]
2	Fraction of mentions	Find out the fraction of @mentions of a user	[5]
3	Average, Maximum, Minimum, Median number of tweets retweets	Find out the average, max, min, and median number of retweets of a user	[11], 5]
4	Total number of times user was mentioned	Find out the total time a user was mentioned	[44]
	URL relation features		
1	Average, Maximum, Minimum, and Median number of URLs	Find out the average, max, min, and median number of URLs of a user	[7]
2	Average, Maximum, Minimum, and Median number of URLs per word	Find out the average number of URLs per word of a user	[7], 10]
3	Fraction of URLs	The ratio of URLs to the number of tweets	[1, 44]
4	Ratio Unique URLs	The ratio of unique URLs to the number of tweets	[40]
5	URL rate	The ratio of URL rate	[44]
6	Number of spam words in screen name	Find out the number of spam words in screen name of a user	[7]
7	Fraction of spam tweets	Find out the fraction of spam tweets of a user	[7]
8	Length Profile name	Count the number of characters on a user profile name	[7]
9	Average spam tweet count	Find out the average spam tweet count of a user	[11]
	Tweets relation features		
1	Number of tweets early morning	Count number of tweets posted between 3:00 am - 6:00 am	[6]
2	Maximum, Minimum, Average, Median, and Standard deviation amount of time between tweets	Maximum, min, average, median, and standard deviation idle time between tweets	[7], 8]
3	Maximum, Minimum, Average, and Median number of characters	Maximum, min, average, and median number of characters from all tweets of a user	[7]
4	Maximum, Minimum, Average, Median number of words	Maximum, min, average, and median number of words from all tweets of a user	[7]
5	Number of Duplicated tweets	Total number of duplicated tweets of a user	[1]
6	Fraction of duplicated tweets	The fraction of duplicated tweets of a user	[10]
7	Maximum number of consecutive words	Maximum number of consecutive words from all tweets of a user	[14]
8	Length Description	The number of characters of description from a user	[15]
9	Number of tweets	Total tweets posted by a user	[7],[15]
10	Average, Mean, Maximum tweet length	Average, mean, and max tweet length of a user	[9, 10, [38]
11	Average time between posts	Find out the average time between tweets	[1]
12	Max idle duration between posts	Maximum idle time between tweets	[11]

13	Tweet similarity - tweet cluster	Find tweet similarity by checking number of unique tweets from a user	[11]
14	Tweet similarity - Cosine Similarity	Find duplicated tweets using Cosine Similarity algorithm	[40]
	Distribution of tweets in 24 hours / Days / Weeks		
1	First, Second, to Eighth set	Number of tweets posted at first set (midnight to 2:59), second set (3:00am - 5:59am), ... eighth set (21:00pm - 23:59pm).	[6]
2	Number of tweets early morning	Number of tweets posted between 3:00 am - 6:00	[6]
	Monday to Sunday relation features		
4	Number of tweets on Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday	Total number of tweets posted on each day of the week.	[15]
5	Maximum, min, median, and average number of tweets on Monday, Tuesday, to Sunday	Maximum, min, median, and average number of tweets posted on each day of the week.	[7]
	Numeric relation features		
1	Maximum, Minimum, Average, and Median number of numeric	Maximum number of numeric on a tweet	[7]
	Weeks relation features		
2	Maximum, Minimum, Median, and Average number of week	Maximum number of tweets posted in one week	[7]

To see exactly the hours spammers tend to be active, we have also added 24 features, which correspond to 24 hours, as shown in Table 3 below.

Table 3 - Invented features based on distribution of tweets in 24 hours

	Invented features derived from Distribution 24-hours features		
	Feature name	**Definition**	**Reference**
1	Number of tweets posted at midnight, 1:00 am, 2:00 am, 3:00am to 23:59pm	Number of tweets posted at midnight, 1:00, 2:00, 3:00 to 23:00	Invented

2.5 Feature Selection

Data almost always contains more information than is needed to create a model and in this research more than 100 features will be extracted, and some of them in the training dataset are very sparse or unrelated, so this will affect the performance of a model if we add them to it. Feature selection is a step of selecting features that are most relevant to a model or removing the unrelated features to improve the accuracy of a system [45, 46] and some of the feature selection algorithms are Information Gain, Chi-Square, Ranker and CFSSubsetEval. The benefit of feature selection is improving the quality of a model and more efficient as less time is needed for feature extraction, building a model and evaluating processes. [10] used Information Gain and Relief methods to find the best five features and [7] used Information Gain and Chi-Square to select the attributes.

In our research, we have collected 172 features. Therefore, we use those feature selection algorithms above to find the optimisation subset of features that are most relevant to the model.

Chi-Square

A Chi-square is used to calculate the worth of an attribute by calculating the chi-square statistic compared to the class. The first step is to find the chi-square statistic value; we need to calculate the expected value as shown below [47].

$$E(r.c) = \frac{n(r) * c(r)}{n}$$

Equation 1 - Expected value format

Where 'r' is the row in the dataset, 'c' is the column, and 'n' equals the corresponding total. In the next step, we could calculate the chi-square statistic value by using Equation 2 below.

Equation 2 - Chi square formula

Where 'O' is the observed value and 'E' is the expected value. Once, we found the chi-square statistic values, then we could use the ranker method to select the features.

CFSSubsetEval

CFS stands for correlation-based feature selection, and the core idea of CFS is that good subset features are highly correlated to the class outcome, but unrelated to each other by evaluating the worth of the attributes through considering the individual predictive ability of each feature [48].

Information Gain

Information Gain values an attribute based on the decrease in entropy after a dataset is split on that attribute. To find the Information Gain of an attribute, we need to calculate two types of entropy: the entropy of the target and the average entropy of attribute. Equation 3 below shows how to calculate the entropy of the target. The term "c" here is the number of classes, and we can denote the proportion of instances with classes i by p_i for i = 1 to "c".

$$\sum_{i=1}^{c} - p_i log_2 p_i$$

Equation 3 - Formula to calculate the entropy of the target

Table 4 - Example of training set for play golf

		Play Golf		
		Yes	**No**	
Outlook	**Sunny**	3	2	**5**
	Overcast	4	0	**4**
	Rainy	2	3	**5**
				14

Assume we have a dataset, as shown in Table 4. So, we have 2 classes: Yes, and No. The number of occurrences of class Yes is 9 (3+4+2) and No is 5 (2+0+3). The value of p_i is the number of occurrences of class i divided by the total number of instances. According to the formula above, we have:

Entropy (Play Golf) = Entropy (9,5) = Entropy (9/14, 5/14) = $-(0.64\log_2 0.64) - (0.36 \log_2 0.36)$ = 0.94.

In the example above we have 3 attributes: Sunny, Overcast and Rainy. Based on Equation 4, the average of entropy for these attributes is calculated as follows:

E (Play Golf, Outlook) = P(Sunny) * E(3,2) + P(Overcast) * E(4.0) + P(Rainy) * E(2,3) = (5/14) * 0.971 + (4/14) * 0 + (5/14) * 0.971 = 0.693

$$\boldsymbol{E(T,X)} = \sum_{\boldsymbol{c \in X}} \boldsymbol{P(c)E(c)}$$

Equation 4 - Average entropy of the attributes

We can calculate the information gain for Outlook by subtracting the Average Entropy of Attribute from Entropy of Target. So the result will be 0.940 - 0.693 = 0.247. We then apply this equation to the other attributes to find the value for them, then select the best attribute based on the highest value.

Ranker

Ranker ranks the attributes according to some evaluation criteria, such as Information Gain, or Chi-Square and returns the top N^{th} attribute; for example, if we set ranking method to 10 for Information Gain, then it will return the top 10 best attributes ordered according to their Information Gain values from highest to lowest [49].

2.6 Evaluation Metrics

In a spam detection system, *Precision*, *Recall*, *F-measure* and *Accuracy* are the common evaluation metrics to evaluate the performance of a system. To calculate those evaluation metrics, we need to find True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. TP means that spam tweets are correctly classified as spam, TN means that normal tweets are correctly classified as normal, FP means that legitimate tweets are classified as spam,

and FN means the number of spam tweets classified as normal tweets [8, 10]. Once we have evaluated the performance of a system, we will use ANOVA, t-test and equivalence testing to compare the performance of that system against the performance of all the other systems.

Precision

Precision is a measure of the classifiers exactness [38]. To calculate the precision, it requires True Positive and False Positive values, and the formula is TP / TP + FP. Also, it is important to be very precise, because precision calculates the occurrence of false positive which are the good tweets classified as spam. Hence, it is important for the precision to be at a high level [50].

Recall

Recall is a measure of a classifier completeness [51]. To calculate the recall, it requires True Negative and False Negative values, and the formula is TN / TN + FN [50].

F-Measure

F-measure is a harmonic means of the *precision* and *recall*; the best result is 1, and the worst is 0 [15]. The formula to calculate F-measure is 2 * TP / (2 * TP + FP + FN).

Accuracy

Accuracy is the percentage of all correctly classified profiles [1]. It is important to use accuracy because when evaluating classifiers, a good balance between precision and recall rates is required, therefore it is necessary to use a strategy to obtain a combined score so accuracy is considered [50] and the formula to find accuracy is ((TP+TN) / (TP+FP+FN+TN))*100.

True Positive Rate

The actual spams correctly classified as spam. It is calculated as TP / (TP + FN).

Time

Here, we look at three criteria, such as input processing time, building model time and classifying time, which may improve the performance of the proposed system. Input processing time means when we extract the features from the dataset, we want to find out how long it takes our system to extract the features needed. Building model time is about the learning time of the classifiers. Classifying time shows how fast a system could detect spammers.

Class Imbalance

With any Twitter dataset, we have a class imbalance problem where there are a lot more normal users compared to spammers in the dataset. [52] counted only 1% of spam profiles from their dataset. Imbalance dataset is common thing [51] and in this case it is more important to accurately predict users as spammers than accurately predict users as legitimate users; the TP rate for

spammers is a better evaluation criterion than Accuracy. Furthermore, the TP rate for spammers is a better criterion than Precision because we would like to predict as many spammers as possible. It is better for us if legitimate users are predicted as spammers rather than the other way around. Out of all the metrics mentioned above, the TP rate is the important metric in this research [51].

T-Test

The T-Test compares two averages together and tells us if they are significantly different from each other. The ratio of T-test is called t-score; this number is telling the difference between the group and the larger t-score, and the greater difference there is between groups. [53]

There are three main types of T-Test:

- An Independent Sample t-test compares the means of the two groups.
- A Paired Sample t-test compares the means from the same group at different times.
- A One sample t-test compares the mean of a single group against a known mean.

In this research we use a paired sample t-test to compares the mean of two models because each model is trained on the same dataset but they are subjected to different subset of features being used for classification.

ANOVA

An ANOVA test is a way to find out if two or more than two experimental results are significantly different from each other. There are two main types of ANOVA: one-way and two-way, and two-way can be tested with or without replication.

One-way is used when you want to test two or more groups to see if there is a difference between them. Two-way without replication is used when you have one group, and you are double-testing that same group. While, with replication, they are used when you have two groups and the members of those groups are doing two or more than two things [54].

In this research we use two-way without replication technique to compare the performance of the models because each model is trained on the same dataset but we have two independent variables: classifiers and the subset of features used to build the model.

Equivalence Testing

We use the equivalence test to determine whether the means for the experimental results are close enough to be considered equivalent [55]. We calculate the maximum interval by adding the average and standard deviation and calculate the minimum interval by substracting the standard deviation from the average. A good system is a system that has high maximum and minimum interval with small standard deviation.

2.7 Algorithms

The extracted features from the dataset will be fed to the algorithms for training the system. According to [23], most of the spammer detection systems have used Support Vector Machine (SVM), Decision Tree (DT), Naïve Bayesian (NB), and Random Forest (RF) as their classifiers. The RF classifier performed well under a small number of the features; the advantage of SVM is that the accuracy does not decrease when we are dealing with many features [8]; DT requires little data preparation; and the NB classifier takes little time for training even with a large dataset [20]. For our research, those classifiers could be considered because they are commonly used in research [23].

Nearest Neighbour Classification

Nearest Neighbour classification is mainly used when all attributes are continuous, and it can be modified to deal with categorical attributes. The ideas of Nearest Neighbour is to classify new data by using the training data that are closest to it. It is usual to base the classification on those of the k nearest neighbour, while the term "k" represents a small integer, such as 2, 3 or 5. To find the best k values, we need to try different k values (e.g. from 1 to 21) for our problems [56]. This method is known as K-NN or K-Nearest Neighbour and the basic of K-NN as follows:

- Find the k training instances that are closest to the class target.
- Take the most commonly occurring classification for these k instances.

To find the "k" training instances, we measure the distances between the unseen data across the training set and sort it from shortest to longest. There are many possible ways of measuring the distances, but the most popular is Euclidean Distance. As shown in Figure 1, if we denote an instance in the training set by (a_1,a_2) and new data by (b_1,b_2), the distance between the points is calculated as Equation 5.

$$\sqrt{(a_1 - b_1)2 + (a_2 - b_2)2}$$

Equation 5 - Distance between the points formula

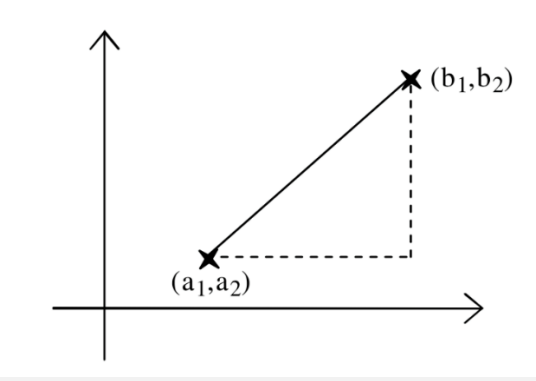


Figure 1 - Euclidean Distance formula

Just for illustration, we will set k to 3, and it is an odd number, so we do not have a tie. After finding the distances between the new data and training set, then we will choose the 3 most similar neighbour (shortest distance) instances to contribute to our prediction.

Decision Tree

Decision tree is an important type of machine learning algorithm used to make a prediction to the target class. The method produces a set of rules by breaking down the complex decision process into a tree structure until the decision is made for the output [57]. This set of rules will be used to perform a prediction on the test set.

Table 5 presents an example of the training dataset to predict a condition that is whether a player is going to play golf or not. In this example, there are 14 objects (5 sunny, 4 overcast, and 5 rain) that were collected through observation of weather conditions. Every object is described by the 4 attributes: Outlook, Temp, Humidity, and Windy, and a classification attribute.

Table 5 - Example of training set for decision tree

Outlook	Temp (°F)	Humidity (%)	Windy	Class
sunny	75	70	true	Play
sunny	80	90	true	don't play
sunny	85	85	false	don't play
sunny	72	95	false	don't play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	don't play
rain	65	70	true	don't play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

One of the important parts of the decision tree is to determine which attribute is the best to be used at each node. To examine the performance of attributes, we use Information Gain as mentioned in Section 2.6 to find the information gain value for every attribute, then we choose the attribute with the highest value.

According to the example above, we can construct a decision tree, as shown in Figure 2.

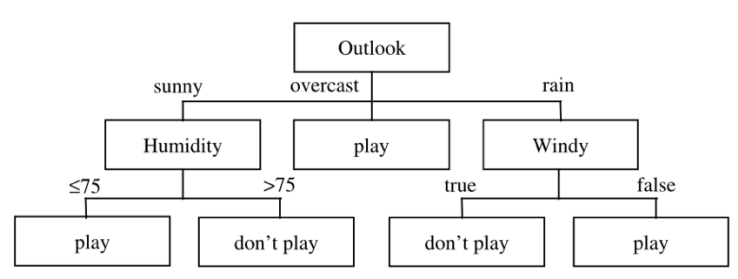


Figure 2 - All the subsets are pure and ready to make a prediction

In order to determine the decision, we go through the set of rules created by the decision tree above. Start at the root node (Outlook). It can be seen that there are three possibilities.

1. If the Outlook is sunny, the next attribute to look at is Humidity. Then, we can see that if the humidity is less than or equal to 75 the decision is to play. Otherwise, don't play.
2. If the Outlook is overcast, the decision is play.
3. If the Outlook is rain, the attribute to consider next is Windy. If it is windy, then the decision is don't play. Otherwise, the decision is play.

Note that, the attribute Temp was not used in this decision tree because its information gain value is low. Thus, if we had the information that the outlook was sunny, with 74% humidity and there was no rain, following the rules above, the decision will be to play golf.

Support Vector Machine

Support Vector Machine is a supervised machine learning that can be used for classification and regression. However, it is more commonly used for classification purposes. The main idea of Support Vector Machine is finding a hyperplane that best separates the training data into two classes (blue and green dots), as shown in Figure 3 [58]. The blue and green datapoints closest to the hyperplane are called support vectors.

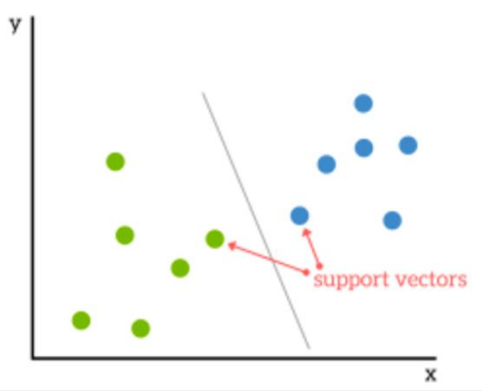


Figure 3 - Example of hyperplane

In two-dimensions we can visualise this as a line and all of the support vectors can be completely separated as follows:

$$B_0 + (B_1 * X_1) + (B_2 * X_2) = 0$$

Equation 6 - Calculation of the hyperplane formula

Where X_1 and X_2 are the input data, while B_0 is the intercept, and B_1 and B_2 are the coefficients that determine the slope of the line calculated by the learning algorithms, such as Linear Kernel SVM. By plugging in the input values into the equation above, we could make a prediction on which class new input data belongs to.

The Linear Kernel SVM model is a line, and the goal of the learning algorithm is to find values for coefficients that best separate the classes [58]. To discover the coefficients' value, we could use sub-gradient descent, and this technique will iterate through the training data and update the coefficients. After the iteration, we will settle on a stable set of coefficients.

Below is the equation to find the output for every iteration update. However, two different types of updates are used, and it depends on the output results.

Output = $Y + ((B_1 * X_1) + (B_2 * X_2))$

Equation 7 - Calculation of every iteration update formula

If the output is greater than 1, it suggests that the training pattern was not a support vector. That means that instance is not involved in calculating the output, so the weight is slightly decreased as follows:

$$b = (1 - 1/t)b$$

Equation 8 - Calculation of the weight if greater than 1 formula

Where b is the weight that is being updated, such as B_1 or B_2, and t is the current iteration. On the other hand, if the output is less than one, then that instance is involved in calculating the output, so the weight is calculated as follows:

$$b = (1 - 1/t)b + 1/(lambda * t)(y * x)$$

Equation 9 - Calculation of the weight if less than 1 formula

The lambda is a learning parameter, and it is often set to a very small value, such as 0.0001 or smaller.

Once, we found the weight (B_1 and B_2), we could plug these results into the equation below and the prediction can be made using the following equation:

$$Y = (B_1 * X_1) + (B_2 * X_2)$$
$$If\ Y > 0, Y = 1$$
$$If\ Y < 0, Y = -1$$

Equation 10 - The prediction formula

Naive Bayes

Naive Bayes is a classification algorithm for binary (two classes) and multiclass classification purposes [59]. It is called Naive Bayes because the calculation of the probabilities for each hypothesis is based on Bayes theorem, as stated in Equation 11 below.

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

Equation 11 - Conditional probability formula

where:

- P(H) is the class prior probability.
- P(E) is the prior probability of predictor (attribute).
- P(E| H) is the probability of predictor given class.
- P(H|E) is the posterior probability of class given predictor.

There are two types of quantities that need to be calculated from the dataset for the Naive Bayes model:

- Class Probabilities P(H) - are the frequency of the instances that belong to each class divided by the total number of instances.
- Conditional Probabilities - are the frequency of every attribute value for a given class value divided by the frequency of instances with that class value.

Let's say we have a dataset as shown in Table 6. It can be seen that, there are 2 classes (class 1 and 2) in the dataset and every class has 5 attributes. So the Class Probabilities for Class 1 and 2 are calculated as

P(H) = P (Class 1) = (class 1) / (class 1+ class 2) = 0.5
P(H) = P (Class 1) = (class 2) / (class 1+ class 2) = 0.5

Where

Class 1 = total number of instances belonging to class 1.
Class 2 = total number of instances belonging to class 2.

Table 6 - Example of a dataset

Weather	Car	Class
1	1	1
0	0	1
1	1	1
1	1	1
1	1	1
0	0	0
0	0	0
1	1	0
1	0	0
0	0	0

Since, we found the class probabilities, the Conditional Probabilities for the weather attribute is calculated as follow:

P(E|H) = P (Weather| Class) = count (weather = 1 $\wedge$ class = 1) / count (class = 1) = 0.8

P(E|H) = P (Weather| Class) = count (weather = 0 $\wedge$ class = 1) / count (class = 1) = 0.2

P(E|H) = P (Weather| Class) = count (weather = 1 $\wedge$ class = 0) / count (class = 0) = 0.4

P(E|H) = P (Weather| Class) = count (weather = 0 $\wedge$ class = 0) / count (class = 0) = 0.6

The $\wedge$ symbol presents the conjunction (AND). Then, we have to repeat the above steps to find the conditional probabilities for Car, and we get:

P (car = 1 | class = 1) = 0.8
P (car = 0 | class = 1) = 0.2
P (car = 1 | class = 0) = 0.2
P (car = 0 | class = 0) = 0.8

Given a Naive Bayes model as below:

$$P(H|E) = P(e_1 \mid H) * P(e_2 \mid H) \ldots P(e_n \mid H) * P(H)$$

Equation 12 - Naive Bayes model

We could make a prediction for new data based on Bayes theorem as follows, where e_1 is the first instance in the dataset. For example, let's use the first instance in the dataset (Weather =1, Car = 1) as shown in Table 6 above, then we get

Class 1 = 0.8 * 0.8 * 0.5 = 0.32
Class 0 = 0.4 * .0.2 * 0.5 = 0.04

In the final step, we can choose the highest calculated value as class 1 (**0.32**), while the smaller calculated value as class 0 (**0.04**). Thus, we predict (Weather = 1, Car =1) "class 1" for this instance, which is correct.

Random Forest

The Random Forest algorithm is a supervised classification, and it can be used for both classification and regression [60, 61]. It works in a similar way to the Decision Tree algorithm; however, it constructs multiple trees at training time. In general terms, the more trees in the forest the more robust the forest is and thus usually it gives more accurate results.

Below is the pseudocode to explain how the Random Forest classifier works. The pseudocode can be split into two stages [61].

- Random Forest creation pseudocode. This stage creates a model to find the class target based on the training set, and it includes 5 steps:
    1. Randomly select a "x" feature from the total number of features.
    2. Among the "x" feature, calculate the node "d" using the best split point (e.g. Information Gain), which means, we are using the randomly selected feature "x" to find the root node by using the best split method.
    3. Split the node from Step 2 into child nodes using the best split method. We keep using the same split method until we form the tree with root node and having a leaf node which is the class target.
    4. Repeat steps 1 to 3 until 'i' number of nodes has been reached.
    5. Repeat steps 1 to 4 to build forests for "n" number of times to create "n" number of trees.
- Pseudocode to perform prediction from the created random forest classifier. In this step, we pass the testing dataset to make a prediction using the trained model in the previous step.
    1. Take the testing dataset and apply the set of rules from the previous stage to predict the outcome or store it.
    2. Calculate the votes for each outcome.
    3. Consider the highest votes to be the best outcome.

2.8 Chapter summary

This chapter outlines the existing techniques to detect spam or spammers for Social Media Services, such as YouTube, Emails and Twitters. In this research, we have aggregated 172 content-based (refer to Table 1), and user-based (refer to Table 2 and 3) features from different systems [4, 6, 7]. There are differences in the results for every existing method, because they used different subsets of features, although some of them used the same one or two features, such as Amleshwaram, Reddy, Yadav, Gu and Yang [21] and [1] used Followers-to-Following-Ratio, but the rest of the features from [21] were different [1], thus we do not know which is the best group of features. The common features in content-based URLs, @mentions, and #hashtags have been utilised by most of the researchers. Additionally, Numbers of followers, or Numbers of followees

are the most common features for user-based. These content-based and user-based features have been expanded into many forms, such as Number of #hashtags per word on each tweet [15] or Fraction of @mention to non-followers [10].

In order to create spam or spammer detection systems, the researchers extract features to build their models. However, some of the features may or may not be relevant to their system [45]. Thus, the feature selection techniques, such as InfoGain and CFSSubsetEval, are used to select the more relevant features when a system utilises many features. Then, the features could be plugged to the classifiers, as mentioned in Section 2.7 to detect spammers. The most common classifiers that have been used to build spam or spammer detection systems are Random Forest, Naive Bayes, Support Vector Machine, Decision Tree, and K-NN [6, 8, 11, 42]. The common evaluation metrics for those systems are Precision, Recall, F-Measure, Accuracy and True Positive. However, due to the class imbalance in the dataset, the best evaluation metric to use is True Positive.

Even when a system produces very good results, for example, [17] achieved 89% in Precision based on Naive Bayes, or achieved 95% in Precision based on Random Forest classifier, over time, the performance of a system will decrease due to the continuous change in the spam statistical features and this is known as the spam drift phenomenon. Therefore, finding a resilient system that is able to detect spam over time is important. In the next chapter, we will explain in detail how to find the best techniques to build an effective, efficient and resilient system to detect spammers.

CHAPTER 3
METHODOLOGY

The objective of this research is to find an efficient, effective and resilient model to identify spammers since they keep evading the existing systems, and it is hard to decide which one is the optimisation subset of features to identify spammers because none of the authors were doing any investigation for this. In the previous chapter, we reviewed how the existing techniques build a spammers' detection system for Twitter in terms of content-based and user-based features, feature selection algorithms, number of recent tweets, classifiers and evaluation metrics.

In order to achieve the objective of this research, we created three research questions as outlined in Section 1.3. The following sections show how we extract and implement each of the content-based and user-based features, and the techniques that we have reviewed in Chapter 2. We then show how do we find the most effective features (RQ1), efficient model (RQ2) and resilient systems (RQ3) called ASDF (Anti Spam-Drift Collection of Features). To facilitate future researchers in this field and to answer the three research questions, we have created a novel tool called WEST (Workbench Evaluation Spammer detection system in Twitter). It not only allows the users to select from the existing 172 features, number of recent tweets, feature selection or classifiers to build their model, but it also allows users to specify new features, feature selection and classifier techniques. Besides that, the users can select certain dates of tweets for feature extraction and evaluate spam drift.

The structure of this chapter is as follows. Section 3.1 shows how to use the WEST tool. Section 3.2 outlines the dataset, the XML tags that we use in this research, and how to prepare the training and test set. Section 3.3 explains how to extract the collected features. Section 3.4 presents the hardware and software have been used for this research. Section 3.5 explains on how we find the ASDF model. Section 3.6 demonstrates the comparison between ASDF model and existing models. Section 3.7 discusses the ASDF model against spam drift problem. Finally, the summary of this chapter is discussed in Section 3.8.

3.1 How to use WEST tool

Presently, no one has done comprehensive, objective comparative studies of all the existing spammer detection systems. To enable people to easily evaluate two or more spammer detection systems against all the common evaluation metrics and investigate how well those systems handle the spam drift problem, we have created a novel tool called WEST. It contains all the required functions to create a spammer detection system such as feature extraction, feature selection, and classification. WEST is written in Java language based on Netbeans platform and uses a collection of algorithms that were imported from WEKA package. Because it is open-source, researchers can extend WEST to include more algorithms or features. In this section, we will show how we can use the program's user interface to prepare the data, set up the features and generate the classification results. This software can be downloaded from https://github.com/kenny6789/WEST.

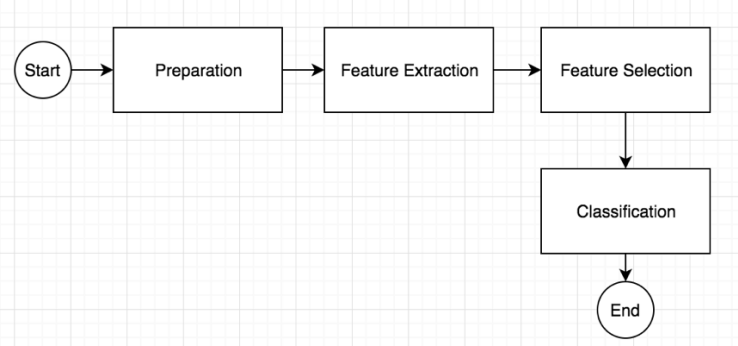


Figure 4 - WEST flowchart

Once, the users download and install the program, there is a screen as shown in Figure 5. By clicking on the Start button, it will navigate them to the main interface of the program, which has four main tabs: Dataset, Feature Extraction, Feature Selection, and Classification, as shown in Figure 6. This is our very first version, and in the future we will expand our program with more functionalities. Therefore in Figure 5 we have many lot of free spaces with the Start button in the middle because in the future we will fill these spaces with other functionalities.

Figure 5 - Click on start button to launch the program

As shown in Figure 6 below, the Dataset tab contains three sections: File Import, Number of Recent Tweets and Extraction Date. In the File Import section, the users must provide data for "Dataset" and "Spam Profile Users".

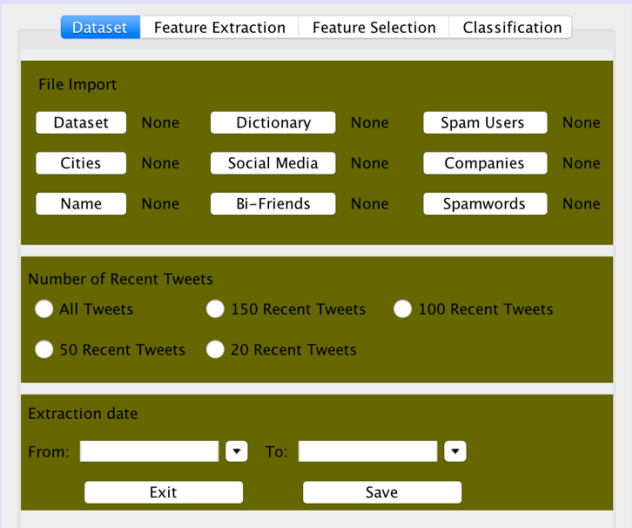


Figure 6 - Main interface of the program

The Dataset must be in XML format, as shown in Figure 7 below. In this XML file, it contains all tags required for feature extraction. Hence, it is important to make sure a new dataset must contain <screen_name>, <followers_count>, <friends_count>, <description>, <tweet_count>, <retweeted>, <text>, and <create_at> tags.

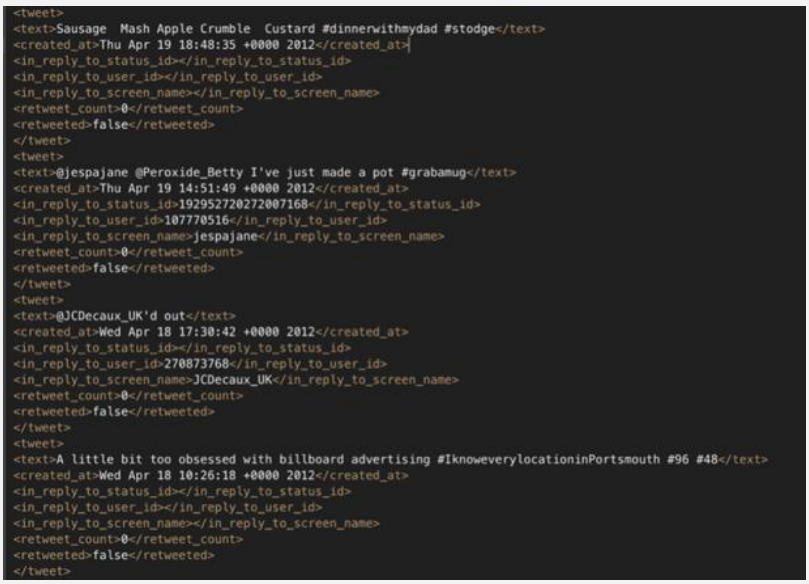


Figure 7 - XML file format

If users would like to extract features that require information from additional knowledge bases, such as spam words as shown in Figure 8 or entity names, users must import all of this information in CSV file format, one CSV file per knowledge base. For example, the spam users requires a CSV file, as shown in Figure 9 containing the list of spammer's profile in the dataset.

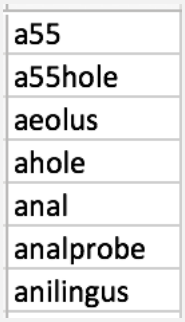


Figure 8 - Spamword list in CSV file

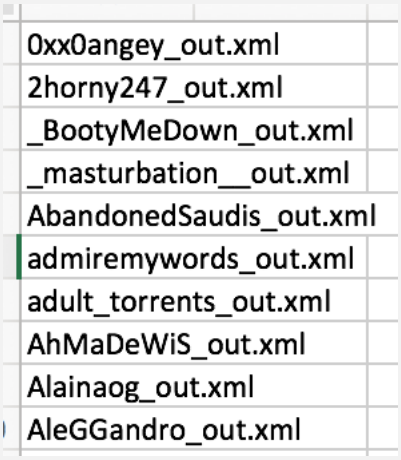


Figure 9 - List of spam profiles stored in a CSV file

The rest of the buttons in the File Import are optional. It depends on the features that would be selected by users. For example, if a user would like to extract "Number of spamwords", then the user must provide a list of spamwords by clicking on the "Spamwords" button. Below is the meaning for the rest of the buttons:

- Places: List of name of the places in the world.
- Name: List of person name in English
- Dictionary: List of valid words in English dictionary
- Social Media Domain: List of social media domain
- Organization: List of organisations or companies
- Spamwords: List of spamwords from social media

While reading the literature review, we found that the researchers were using a different number of recent tweets. Therefore, we have created several options for the users to choose from. In this section, the users could choose the desired number of recent tweets: 200 recent tweets, 150 recent tweets, 100 recent tweets, 50 recent tweets, and 20 recent tweets and by default WEST will start at 20 recent tweets.

Furthermore, we also created a filter to let the users select tweets in a certain date range. This filter will take all the tweets in the specified range for feature extraction. The purpose of this function is

to allow users to test their spam drift; for example, if the dataset contains tweets from 2012 to 2014, the users could extract all the tweets in 2012 and 2013 for training and testing their model and assuming they found a good model, then they could test their model again on the tweets in 2014 to find out if their model is performing well against spam drift or not. It is important to make sure that the dataset have tweets in the specified range as otherwise there will be no tweets extracted from the dataset. The users must click on the Save button to save all the settings for the Dataset tab.

Once, the preparation is done, the users could select content-based and user-based features that they want to extract from the Feature Extraction tab. In Figure 10, we have provided 10 systems for the users, and each of those system is already specified with features based on their literature. For example, if the users choose system 2, and clicked on "Apply" button, the content-based and user-based features that are specified for that system will be automatically selected, as shown below. The existing systems which have been implemented in WEST are [1], [10], [44], [6], [15], [40], [43], [7], [11], and [14], and because the authors did not have the name for their systems, therefore, we just called them System 1 to System 10, respectively as shown in Figure 8.
Also, we have a "New System" option that allows users to create their own subset of features, by default this option will select all the content-based and user-based features. Please note that, all the extracted features will be saved in arff file format as shown in Figure 13.

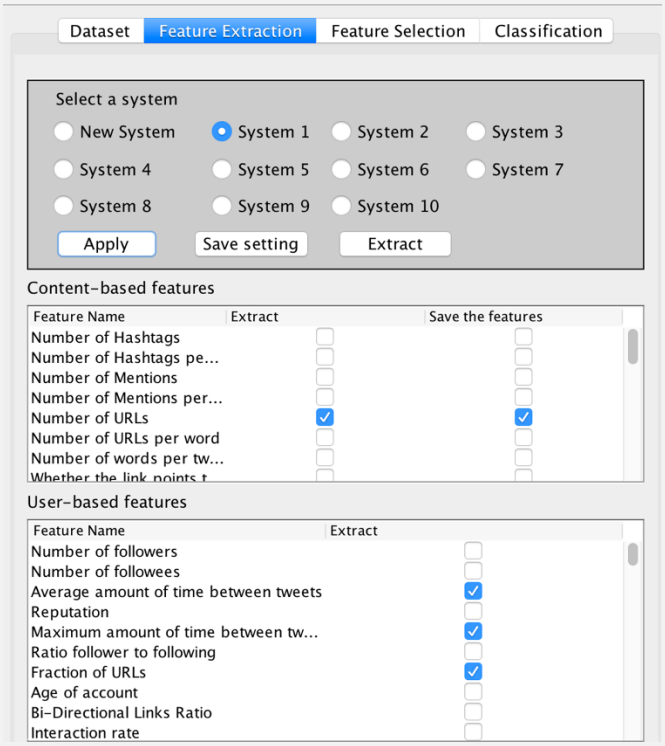


Figure 10 - Feature extraction tab

Because some of the user-based features are derived from content-based, therefore, it is necessary to make sure that we must select the right derivation for that user-based feature. In Figure 11, I have selected "*Maximum number of hashtags*", and the message showed that we have to select

"*Number of Hashtags"* to extract *"Maximum number of hashtags"*. In case the users do not want to include "*Number of Hashtags"* in their arff file, they could untick the box from the "Save the features" column in Content-based features. This means that *"Number of Hashtags"* will be extracted from the dataset but its value will be discarded after it has been used to calculate the *"Maximum number of hashtags"*. *"Number of Hashtags"* will not be included as one of the attributes is in the arff file. Once they have selected the set of features that they would like to extract from the dataset, they could click on the "Save button" to save all the settings in this process.

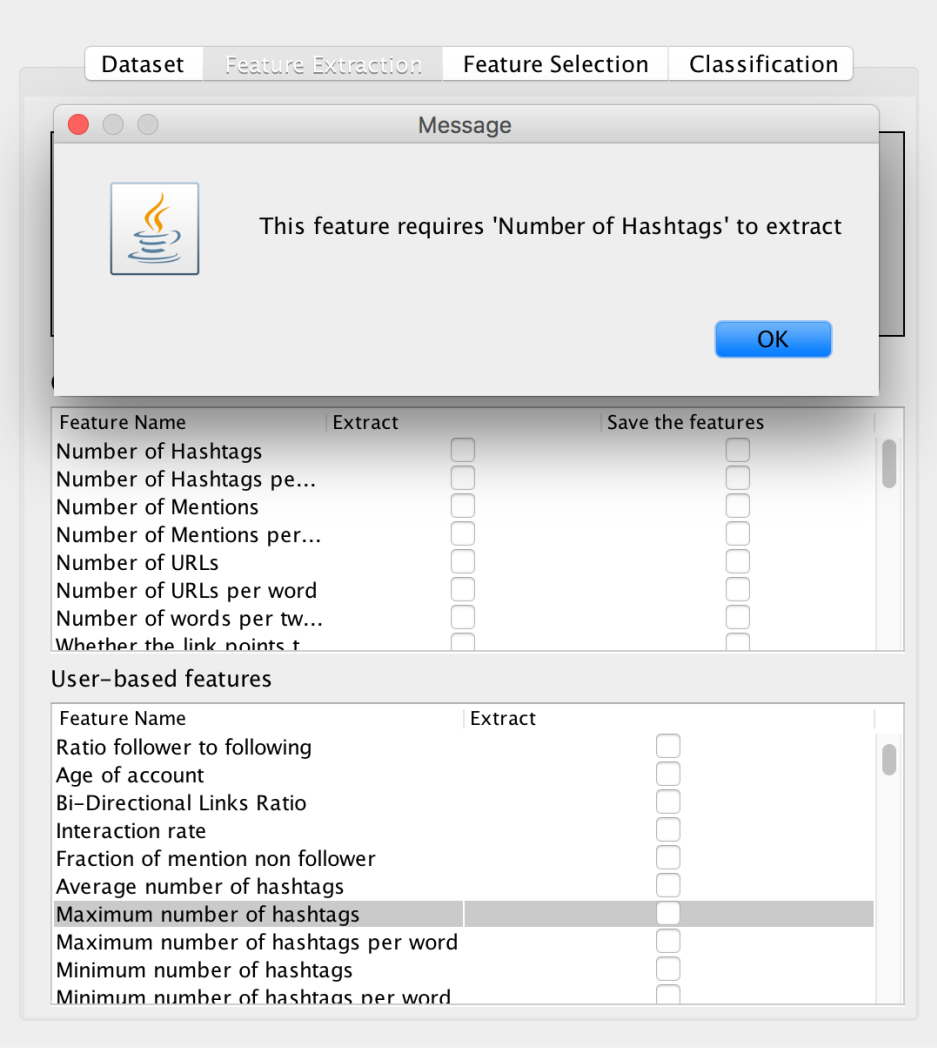


Figure 11 - Maximum number of hashtag features is derived from the number of hashtags

Once the user clicked on the "Extract button", the system will ask for the location to save the three arff files that contains the extracted content-based only, user-based only and combination (content-based and user-based) features respectively, as shown in Figure 12 below. In most of the cases, only the combination arff file will be used by the users, but we keep the content-based and user-based features to enable users to check if any problems occur.

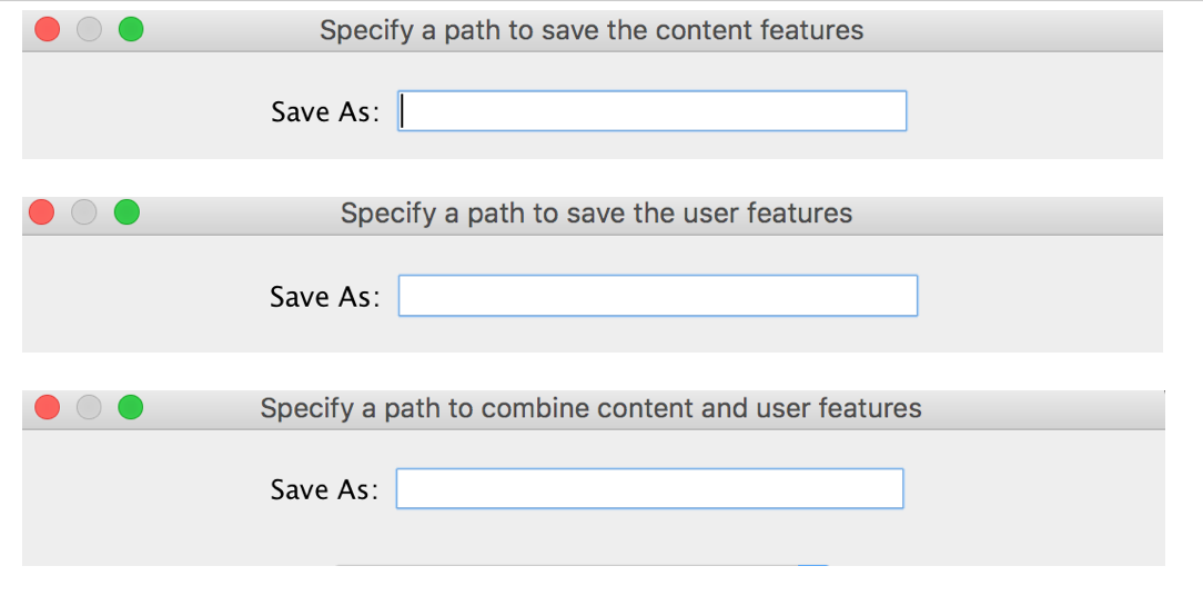


Figure 12 - Specify the paths for arff files

Figure 13 presents an example of the output arff file produced after the “Feature Extraction” process. It contains the heading, the attribute names and datatypes, and the data. The heading contains the name of the system that we have selected from "Feature Extraction" process. The attribute contains the name of the features that were selected before, and the data type for that feature. Lastly, the information or features that are extracted from the XML files are converted into a meaningful format under the @data tag for the classifier's algorithms. Each of the lines of information is assigned with a class which is ham or spam.

```
@RELATION NewSystem

@ATTRIBUTE MaximumNumberOfHashtagPerWord numeric
@ATTRIBUTE MaximumNumberOfWordsPerUser numeric
@ATTRIBUTE class {ham,spam}

@DATA
0,19,ham
0.11111111,10,ham
0,28,ham
```

Figure 13 - Arff file format

Figure 14 shows the feature selection tab. When the users would like to find out what are the optimisation subset of features from the extracted features in Feature Selection step, they could use this function to find it. To use this function, WEST requires an arff file (shown in Figure 13), and the user needs to choose a feature selection algorithm then click on the "Apply button" to find the optimisation subset of features. In Figure 14, we showed an example of the returning list of attribute ranking based on their information gain values. Please note that, this step is optional, and only suitable to be used when we have many features.

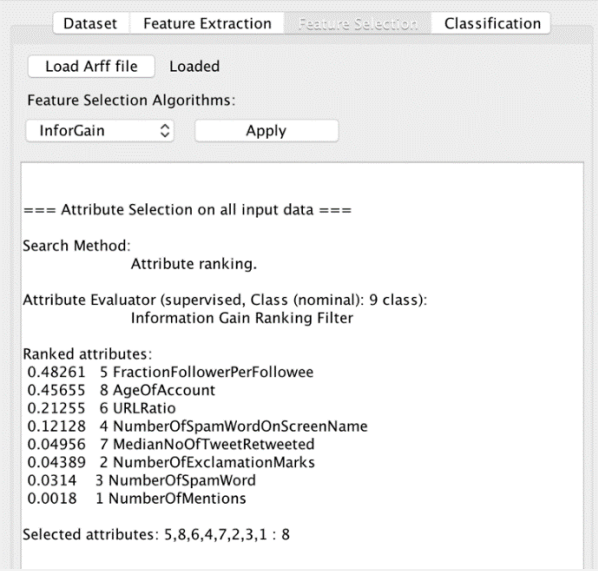


Figure 14 - Feature selection tab used to find the optimisation subset of features

The classification tab is used to evaluate the spammer detection systems based on the provided training and testing arff files. As shown in Figure 15, there is a button called "Load arff Files", and this is where the users import their training and testing arff files. WEST will recognise the training and testing set based on its name. Thus, we must have the word "Training" and "Testing" in the names of the training and testing files respectively. For example, TrainingToFindSpammers.arff for the training set and TestingToFindSpammers.arff for the test set.

After loading the arff files, users have to select a classifier algorithm. In Figure 15, Random Forest classifier is selected. Then, the users could choose one, two or all the evaluation metrics and click on the "Apply button" and the results will display below the Classification Results section, as shown in Figure 15.

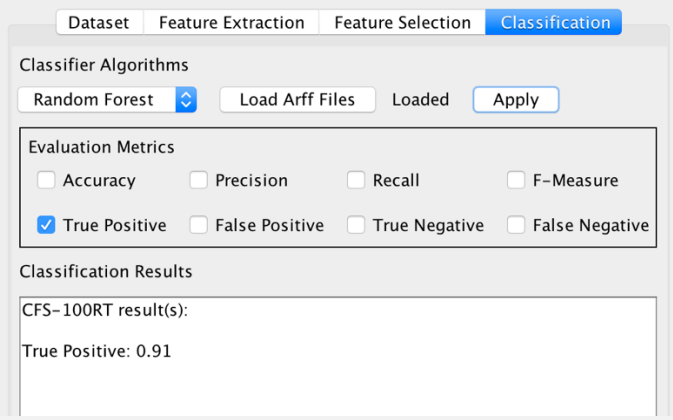


Figure 15 - TP result of CFS-100RT subset of features based on Random Forest classifier.

3.2 Dataset

For this research, we will use the dataset from [11] for training and finding a collection of effective and efficient subset of features, while [39] are used to test the model for RQ3, because these are the only dataset we have found that contains the tweets and user profiles in their raw original format. It allows us to extract all the features from Tables 1, 2 and 3. Other datasets, such as [7], only have the extracted values for each feature they use; therefore, we could not use them.

The dataset [11] has 7,549 profiles, already separated into spam profiles (315 profiles) and validated profiles (7,234 profiles). There is still some additional information required by some of the features listed in Tables 1 and 2 that was not available on the dataset, such as *Age of Account* and *Bi-Directional Links*. To acquire this information, we use Twitter4J [62] to get the data from Twitter based on the <screen_name> or <id> tags of each user in the dataset. Because some of the profiles in the dataset did not have that information or they were simply not available on Twitter anymore, we had to remove them from the dataset. Thus, we are left with 1,729 profiles (315 spam and 1414 ham).

We choose the dataset [39] for testing spam drift because this dataset has the latest tweets in 2015, while [11] is containing tweets from 2012 to 2013. Tweets in dataset [39] are available in raw data format, which means we can do the feature extraction process, and it contains around 4800 ham and 6700 spam profiles.

We splitted the [11] dataset into a training set and a test set. The training set contains 70% of the validated and spam profiles, and the test set involves 30% of the rest of the validated and spam profiles. We chose 70% for the training and 30% for the test set because this is often used by many researchers [63]. We used the whole [39] dataset as our second test set for evaluating how well the systems at handling the spam drift phenomenon.

The dataset [11] is in XML format, and the information is separated into different XML tags, as shown in Figure 16. However, the dataset [39] is in CSV format, because WEST can only accept XML format, we have converted the dataset in XML format. Each node contains several tags, such as <text>, <created_at>, <retweeted> or <in_reply_to_screen_name> and those tags are used to obtain the content-based features, while other tags will be used for user-based features, such as <followers_count>, or <statuses_count>. For example, to find *Number of characters* content-based feature, we must first extract the string "Missed the bus at one stop picked it up two avenues later. I can be a very fast walker" from <text> tag then calculate the number of characters in that string. For a user-based feature like *Number of Followers*, it can usually be extracted directly from reading the appropriate tag like <followers_count> tag. Common features such as #hashtag, @mention and URLs are extracted from <text> tag.

In this research, we used data extracted from some XML tags mentioned in Section 3.4.1 and 3.4.2 below. However, we only do the cleaning process for the <text> tag because this is the main tag containing many text characters, while the other tags just contain a small amount of characters with relatively very clean text. For example, we will convert "Let#39;s work a poco??" to "Let's work a poco??" by replacing "#39;" to a single quote. For the [39] dataset, we do not need to perform the cleaning process because this dataset is already cleaned by the author.

Figure 16 - All the XML tags from a user's profile

3.3 How to Extract the features

As mentioned in Section 2, there are two types of features, content-based and user-based, and those features will be extracted from the dataset mentioned in Section 3.2. However, because some of the user-based features are derived from content-based features, it is therefore necessary to extract the content-based features before the user-based ones. The following sections explain how to extract all the features mentioned in Tables 1, 2 and 3.

In this research, we will extract all the collected features from the various authors mentioned in Section 2 by ourselves. Because some authors do not have the code to their systems available, we have extracted the features based on what was written in their paper but we cannot be 100% certain that what is implemented in WEST exactly matches the author's actual implementation. Below, it demonstrates how each content-based and user-based feature is extracted from the dataset.

3.3.1 Content-Based features

As mentioned in Section 3.1, the<text>, <retweeted> and <created_at> tags are used to extract the content-based features. In this research, we used six CSV files "*Name"* [64] "*Cities"* [65], "*Organization*" [66], "*Social media domain*"[67], "*Dictionary*" [68], and "*Spam words*" [69] as our knowledge base files. The "name" CSV file contains 5,494 names. The "cities" CSV file contains 86 of the largest cities in the world. The "organization" CSV file contains 2000

organisation names in the world. The "social media domain" contains 141 social media websites. The "dictionary" is an English dictionary containing 47,161 words. Finally, "spam words" contains 723 bad words blacklisted on Facebook. We will use those CSV files to extract *Tweets containing Places, Tweets containing oganizations, or Percentage of words not found in the dictionary*. Next, we will explain how WEST extracts the features mentioned in Table 1.

- *Common features*

The feature *Number of URLs* (content-based number 1) is extracted by checking if a tweet contains "http" and "www" or not; if there is a "http" or "www" in a tweet then the *Number of URLs* is increased by one; for example, "*@MoniqueFrugier Hi I supposed that you'd be pleased with this. Good luck http://wpnewsreport.com/googlenews/?=mtiy*" and the number of URLs for this example is 1.

For *Number of URLs per word* (content-based number 2), we did the same step as *Number of URLs,* but we also counted the *total number of words* in a tweet, then took the *Number of URLs* divided by the *total number of words.* For example, "Disasters call federal honours into question http://t.co/earCf76", and this string has six words and one URL so the *number of URLs per word* is 0.16 (we do not count a URL as a word). These steps were applied to *the Number of Mentions* (content-based number 7), *Number of Mentions per word* (content-based number 8), *Number of Hashtags* (content-based number 4), and *Number of Hashtags per word* (content-based number 6).

Whether the links point to a social media domain (content-based number 3) is extracted by using the split() method to get the URL part from a text. Then this link was checked against the "social media domain" CSV file. If the extracted link contains any social media name in the CSV file, it is considered as pointing to a social media domain. For example, in "Pay special attention at 2:54 http://www.youtube.com/watch?v=yAQoLZKfYXc", this tweet navigates to YouTube, which is a social media domain.

- *Special Characters*

Special characters include exclamation marks, question marks, ampersand, at sign, or dollar sign. However, for this project, only exclamation marks and question marks were considered as special characters because [15] only extracted those features; thus, we did not attempt to extract other special characters.

To find *the number of question marks* (content-based number 10) from a tweet, we checked every character from a tweet and counted the number of question marks in that tweet. For example, in "*@dvarimbealma ????=?????*", there are nine question marks and it is applied to find *the number of exclamation marks* (content-based number 9) as well.

- *Retweet*

In Twitter, retweet is defined as "RT @username". To find *Number of Retweet* (content-based number 13), we broke a string into a substring, then we checked if the string started with "RT", and we did not care about how many "RT" were in a tweet because we counted it as one "RT". For example, in "*RT @Leslie_Lou109: RT @parker_story: I can't write papers unless they're due in a matter of hours*", this string has one retweet.

Another feature similar to *Number of Retweet* is *Whether the tweet is retweeted* (content-based number 12). In the XML file, there is a tag called <retweeted> and this tag contains a Boolean value. If the return value is true, then it is a retweet, otherwise it is not a retweet. However, our training and testing sets are in numeric format; therefore, we converted the return value to a numerical value, for example, false is 0 and true is 1.

- *Numeric*

To find *numeric characters* (content-based number 11), we broke a string into a substring then we checked every character to find the numeric characters. In this research, we did not count tens or hundreds because the name of this feature is *numeric characters* and the author who proposed this feature did not explain well whether to count a whole number or just count a numeric character. Therefore, we decided just to count the number of numeric characters. For example, *"...what was supposed to be a 30 minute drive to work took nearly 2 hours. #ihatetraffic",* so this string contains three numeric characters.

- *Tweets/Words*

To find *Number of words* (content-based number 26), we used StringTokenizer class and set the delimiter as whitespace to break a string into tokens. Then we counted the total number of words from the token. For example, *"sorry couldn't help it",* contains four words.

To find *Number of Spam Words Per Word* (content-based number 28), we broke a string into tokens then checked every word in the token against the "spam words" CSV file to get the *total number of spam words.* After that, we took the number of spam words divided by *the Number of Words* to get *Number of Spam Words Per Word.* For example, in *"...work is stupid slow. gonna be another long day",* the word "stupid" is a spam word, so *number of words* is nine and *total number of spam words* is one. *Number of Spam Words Per Word* is 1/9.

To extract *Number of consecutive words* (content-based number 15) feature, in our system we considered a consecutive word as containing two words that are all alphabetical characters, except for two cases like the pronoun "I" or article "A", which we still consider as a word. For example, in "*have you any idea why a raven is like a writing desk?*", "have you" is one consecutive word and "you any" is another consecutive word and so on.

To find *Number of characters* (content-based number 16), we looped through a text and counted the number of characters in that text, and we did not count a whitespace as a character because [15] used the whitespace as a feature to count to find out how many whitespaces were on a tweet.

For example, in *"I wish I could make a bomb that would kill every snake in the world"*, this string contains 53 characters.

To find *Number of whitespaces* (content-based number 17), we looped through a text and checked every character, to see if it was a space, then we considered it a white space. For example, in "*I wish I could make a bomb that would kill every snake in the world*", this string contains 14 white spaces.

To find *Number of Capitalization words* (content-based number 18) and *Capitalization words per word* (content-based number 19), we used StringTokenizer to break a text into tokens then checked every word in the token to see if the first character was a capital or not. After that, we counted the total number of words in the token then took the number of capitalised words divided by the total number of words to get *Capitalization words per word.* For example, in *"I hate my Blackberry"*, the *Number of Capitalization words* is two, and the total number of words is four *so Capitalization words per word* is 0.5.

To find *duplicated tweets* (content-based number 20), we applied the *Min Distance* algorithm, which is a way of quantifying how similar two strings are by counting the minimum number of operations required to transform one string into the other [70]. We created a minDistance() method based on the algorithm, which takes two parameters (text1 and, text2). If a return value is 0, it is absolutely duplicate and 1 or greater than 1 means two texts are different.

To find *Percentage of words not found in dictionary* (content-based number 21), we used the downloaded *Dictionary* [68] with 47,161 words. Then we used HashMap to store the words in the dictionary and check every word from a text against this HashMap. If a word does not exist in the HashMap we considered it not found in the dictionary. The formula took the number of words not found in dictionary divided by total number of words then multiplied by 100.

For example, in *"...work is stupid slow. gonna be another long day"*, the word "gonna" is not in the dictionary, so the number of words not found in dictionary is one and the total number of words is nine, so the *percentage of words not found in dictionary* is (1 / 9) * 100.

To find *Tweets contain Social Media Domain*, we used the "socialmediadomain" CSV file and checked if a tweet contains any social media domain. For example, in *"Pay special attention at 2:54 http://www.youtube.com/watch?v=yAQoLZKfYXc"*, this tweet contains one social media domain which is Youtube.com. This also applied to *Tweets contain Places* (content-based number 22), *Tweets contain Organization* (content-based number 23) and *Tweets contain Names* (content-based number 24).

To find *Time of Publication* (content-based number 29), we used the information from <created_at> tag in XML file and the format of this tag was “Sat Apr 21 03:47:18 +0000 2012”. Then, we removed the date, month, year, minutes, seconds, and just kept the hour so after the removal, the data would be “03”. For this feature, we only kept the hour, because we did not really care about the exact time and also [42] did not explain this feature well.

3.3.2 User-based features

Most of the user-based features are based on the content-based features, for example, to extract *Maximum number of #hashtag per tweet* we have to get the number of #hashtag. Additionally, there are some user-based features that do not need content-based, such as *Length of profile name, Number of follower, and Number of followee.*

In this research, we studied 145 user-based features and created several methods to extract those features, for example, getAverageNumber() can find the average number of #hashtag, @mention, or URL by passing the required data, and Table 7 shows the functions used to extract all the user-based features from Table 2.

Table 7 - Function name and feature number

Function Name	***Feature Number***
Max, Min, Median, Average, Standard Deviation	
1.1 getAverageNumber()	9, 18, 28, 51, 55, and 145
1.2 getMaxNumber()	10, 19, 25, 30 , 44, 49, 53, 59, 64, 68, 104, 109, 114, 119, 124, 129, 134, 138, and 142.
1.3 getMaxNoXPerWord()	11, 31
1.4 getMinNumber()	12, 20, 26, 32, 45, 50, 54, 105, 110, 115, 120, 125, 130, 135, 139, and 143.
1.5 getMinNoXPerWord()	
1.6 getMedianNumber()	14, 15, 21, 27, 34, 36, 47, 52, 56, 106, 111, 116, 121, 126, 131, 136, and 141.
1.7 getMedianNoXPerWord()	13, 33
1.8 getStandardDeviation()	48
User Profile	
1.1 Number of spamword on screenname	39
1.2 Length of profile name	41
1.3 Ratio follower to following	4
1.4 Length of description	60
1.5 Number of tweets	61
1.6 Age of account	5
1.7 Bi-directional Links	6
1.8 Fraction of mention to non follower	8
Time	
1.1 getTimeBetweenTweets()	44, 45, 46, 47, and 48.
1.2 Number of tweets in early morning	43
1.3 Distribution of 24-hours (in 0-23)	From 70 to 93
1.4 Distribution of 24-hours (in 8 sets)	From 94 to 101
1.5 Number of tweets posted on Mon – Sun	103, 108, 113, 118, 123, 128, and 133
1.6 getMaxMinAvgMedDay()	104-107,109-112,114-117,119-122,124-127, 129-132 and134-137
1.7 getMaxMinAvgMedWeek()	142, 143, 144, and 145
1.8 Average time between posts	63
1.9 Max idle duration between posts	64
Tweets	
1.1 Number of unique duplicated tweets	57
1.2 Average tweet length	62
1.3 Average spam tweet count	42
1.4 Tweet similarity - tweet cluster	65
1.5 Tweet similarity - Cosine Similarity	66
Retweet	
1.1 Average retweets per tweet	23
1.2 Interaction rate	7
1.3 maxNoofTweetRetweeted	25
1.4 minNoofTweetRetweeted	26
1.5 medianNoofTweetRetweeted	27
URL, @Mentions, #Hashtag	
1.1 Average URL count	28
1.2 URL ratio	38
1.3 Ratio Unique URLs	37
1.4 Average Hashtag count	9

1.5 Hashtag ratio	17
1.6 Fraction of #hashtag, @mention, and URL	16, 22, and 35
1.7 Total number of users mentioned	24

1) getAverageNumber(total#PerTweet ,totalTweetsContain#)

This method takes two parameters, the total number of #hashtags from every tweet (total#PerTweet) and the total number of tweets containing #hashtag (totalTweetsContain#). The total#PerTweet is the total number of hashtags from all tweets, and the totalTweetsContain# is the total number of tweets containing hashtags. We do not count any tweets that do not contain hashtags.

Average number of #hashtag = total#PerTweet / totalTweetsContain#
The example below illustrates how to find total#PerTweet and totalTweetsContain#
tweet-1: *Just broke into my own house. #brokenwrist #whoneedsalocksmith*
tweet-2: *#Bestsong of all time #forever http://t.co/ORTab1W0 #Whitney*
tweet-3: *Bought a new kitchen today*

So the total#PerTweet is five, because tweet-1 contains two #hashtags and tweet-2 contains three #hashtags and the totalTweetsContain# is two because only tweet-1 and tweet-2 contain hashtags. The average number of #hashtag = 5 / 2.

2) getMaxNumber(arraylist#hashtag)

This method takes one parameter which is an arraylist. For example, we put all the numbers of #hashtag from every tweet to an arraylist arraylist#hashtag. To get the maximum number from the arraylist, we use the Collections.max(arraylist#hashtag) method, and this returns the maximum element of the given collection.

For example:

Tweet Number	Number of Hashtag
1	6
2	8
3	9

So the arraylist holds {6,8,9}, and the maximum value is 9.

3) getMaxNoXPerWord(arraylist#hashtagperword)

The term of "X" could be #hashtag, @mention, or URL. For example, to extract *Maximum number of #hashtag per word,* we have to extract the *number of #hashtag per word* feature first. After that, we insert all the numbers of *#hashtag per word* to the arraylist. To get the maximum number of *#hashtag per word,* we use the Collections.max(arraylist#hashtagperword) method to return the maximum element of the given array.

For example:

Tweet Number	NoOfHashtagPerWord
1	6
2	6
3	8

So the arraylist holds [6,6,8], and the maximum value is 8.

4) getMinNumber()

This method requires an arraylist parameter. Instead of using Collections.max(), we used Collections.min() to get the minimum element from the arraylist.

5) getMinNoXPerWord()

Similar to getMaxNoXPerWord() method, the getMinNoXPerWord() finds the minimum value.

6) getMedianNumber()

For example, to find a median number of #hashtag of a user, we added all the number of #hashtags to an array then sorted it in order. The median number of#hashtag is the value located in the middle of the array.

For example:

Tweet Number	NumberOfHashtag
1	6
2	9
3	8
4	2
5	7

After inserting and sorting all the *number of #hashtag*, the arraylist becomes {2, 6, 7, 8, 9} and the median value is 7.

7) getMedianNoXPerWord()

Similar to the getMaxNoXPerWord() method, the getMedianNoXPerWord() finds the median value.

8) Fraction of #hashtag, @mention, and URL

To find a fraction of #hashtag, we have to find the number of tweets containing #hashtag, and it is content-based features. After that, we divide the number of tweets contain #hashtag by the total number of tweets.

The fraction of @mention, URL, DuplicatedTweets and Spam tweets was extracted in the same way. However, this function is different to getAverageNumber() because *Fraction of #hashtag* and getAverageNumber() did not use the same values.

Fraction of #hashtag = the number of tweets containing #hashtag / the total number of tweets.

9) getTimeBetweenTweets()

This method checks the time between two tweets. The time of a tweet is extracted from <created_tag> tag. To find the time between two tweets, we took the time from tweet1, then subtracted the time of tweet2 and, for this research, the time between tweets is in minutes.

For example*:*
Tweet1:
<text>@MustLoveCyrus i hope u dont mind my username lol </text>
<created_at>Wed Jan 27 00:05:53 +0000 2010</created_at>

Tweet2:
<text>still selena but different pic</text>
<created_at>Wed Jan 27 00:00:14 +0000 2010</created_at>

The time between two tweets is five minutes.

10) Number of spamwords in screen name

This feature represents the total number of spam words that appear in the screen name. We got a user's screen name from <screen_name> tag in the XML file then checked it against the list of spamwords in CSV format to find the number of spam words.

For example, in *"work is stupid slow. gonna be another long day"*, the word "stupid" is a spam word in *spamwords* CSV file, so this text contains one spam word.

11) Length of profile name

To get this feature, we used the screen name from <screen_name> tag then counted the number of characters in the string. For example, a screen name *"Fox_McCloud_"* has a length of 12 characters.

12) Number of tweets in early morning

For this feature, we considered early morning as the time between 3:00am and 6:00am. To get this feature, we took the information from <created_at> tag and formatted it into "H:mm:ss". After that, we created two ranges, 30,000 and 60,000. To extract this feature, we removed the icon ":"

from “H:mm:ss”, for example, *“3:42:59”,* after the removal, becomes “34259”, then we checked if “34259” was in the range between 30,000 and 60,000.

13) Number of unique duplicated tweets

In this system, we defined “duplicated tweet” as two tweets that are exactly the same.
To get this feature, we needed to use Hashset and Min-Distance algorithm. First, we checked a pair of texts with a min distance algorithm, and if it was absolutely duplicated, which is 0 distance, we added it into the hashset. Finally, we counted the total number of elements in Hashset to get the number of unique duplicated tweets.

For example, we have three tweets as below:
tweet1*: I wish I could make a bomb that would kill every snake in the world*
tweet2*: I wish I could make a bomb that would kill every snake in the world*
tweet3: *Bought a new kitchen today*

The *number of unique duplicated tweets* is one because tweet1 and tweet2 are exactly the same.

14) Distribution of 24-hours (in 0-23)

To find the number of tweets posted from 0-23, we used the feature *Time of publication.* For example, to find the number of tweets posted at 3:00 am, we counted all the tweets that have a "3" value from *Time of publication.*

For example:

Tweet Number	Time of Publication
1	3
2	9
3	6
4	8
5	3

So the number of tweets posted at 3:00 is two, as tweet number 1 and number 5 were posted at 3:00.

15) Distribution of 24-hours (in 8 sets)

We did the same step as “Distribution of 24-hours (in 0-23)” to get the hours. Instead of counting the tweets in a one-hour range as in *Distribution of 24-hours (in 0-23),* we divided 24 hours into eight sets; for example, the third set was from 6:00 to 8:00, then we counted the number of tweets posted in this range.

For example:

Tweet Number	Time of Publication
1	3
2	9
3	6
4	8
5	3

So there were two tweets posted in the third set because tweet numbers 3 and 4 were posted at 6:00 and 8:00.

16) Fraction follower to following

To get this feature, we took the number of followers and followees from the XML file. Then we took the number of followers divided by number of followees.
FractionFollowerPerFollowee = number of followers / number of followees.

17) Number of tweets posted on Mon – Sun

For this feature, we used Monday as an example. In the XML file, we took the data from <created_at> and we had a string, "Mon Apr 21 00:58:44". Then we removed everything from the string except "Mon". After that, we just counted the number of "Mon". Also, this was applied to Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday.

18) getMaxMinAvgMedDay()

This function can be used to find the maximum, minimum, average and median number of tweets posted on Monday in different weeks.

First, we started at the lastest tweet from a user and found the range of a week for that tweet. Then we counted the total number of tweets posted from Mon to Sun that week, and after that we kept the findings for other weeks and created a CSV file to save all of it.

Figure 17 shows the number of tweets posted in two weeks by a user. Then we counted the number of tweets posted every week and created a CSV file to store it, as shown in Table 8.

Week	Text	Post at
	check out this article I made 270 today http://t.co/hnWMCEh	Mon Jul 11 17:51:33 +0000 2011
	Local single mom makes 2300/w. online. read more at http://t.co/eEjkLd1	Mon Jul 11 21:29:20 +0000 2011
	check out this article I made 280 today http://t.co/gnnnxep	Wed Jul 13 18:48:01 +0000 2011
	makes almost 5500/month working part time. check out this article http://t.co/OX6kALR	Thu Jul 14 19:52:55 +0000 2011
	check out this article I made 280 today http://t.co/GNaoBxZ	Fri Jul 15 02:03:13 +0000 2011
	check out this article I made 350 today http://t.co/XgLi3bg	Sat Jul 16 20:07:57 +0000 2011
1	check out this article I made 350 today http://t.co/TNDq9C3	Sun Jul 17 02:16:05 +0000 2011
	Local moms earns 387/hr Online. find out how here http://t.co/dcnyurN	Tue Jul 19 20:34:37 +0000 2011
	check out this article I made almost 350 today http://t.co/1cPCEgy	Wed Jul 20 04:37:39 +0000 2011
	check out this article I made 400 today http://t.co/s19lGDh	Wed Jul 20 12:23:30 +0000 2011
2	check out this article I made 300 today http://t.co/5SpDeiL	Wed Jul 20 19:37:44 +0000 2011

Figure 17 - Number of tweets posted from Monday to Sunday in a week

To get the maximum number of tweets posted on Monday, we added the number of tweets posted on Monday of weeks 1 and 2, in Table 8, to an arraylist then passed the arraylist to the getMaxNumber() method to find the maximum value, so the maximum number of tweets posted on Monday was two.

Table 8 - A created CSV file for storing number of tweets posted every week

Week	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	2	0	1	1	1	1	1
2	0	1	3	0	0	0	0

To find the minimum, average and median number of tweets, we passed the arraylist to the getMinNumber(), getAverageNumber() and getMedianNumber() methods.

19) getMaxMinAvgMedWeek()

To get this feature, we used the CSV file mentioned in Table 8. Then we added up all the values from Mon to Sun per week and put the result in an arraylist. After that, we passed the arraylist to getMaxNumber() to get the maximum tweets posted in a week.

For example, Table 8 has two weeks and the number of tweets posted on Monday to Sunday was 2+0+1+1+1+1+1 = 7 tweets, and the second week was 0+1+3+0+0+0+0 = 4 tweets. Then we put four and seven to the arraylist and passed the arraylist to getMaxNumber() method, so the maximum number of tweets posted in one week was seven.

To get minimum, average or median values, we changed the getMaxNumber() method to getMinNumber(), getAverageNumber() or getMedianNumber() methods.

20) Length of description

The *Length of description* can be extracted from the <description> tag in the XML file. We counted the total number of characters from this tag. In this research, we considered a whitespace as a character for this feature. For example, <description>*Fun be Happy dance*</description> has 18 characters.

21) Number of tweets

To extract *Number of tweets* feature, we used <tweet_count> tag, for instance, <tweet_count>41</tweet_count> and this tag contains the total number of tweets posted by a user.

22) Average retweet per tweet

We have extracted the *Number of retweets* feature. To find *Average retweet per tweet*, we added all the *Number of retweets* to find the total number of retweets. After that, we divided the total number of retweets by the number of tweets to get the average retweet.

For example:
Tweet-1:*RT @Leslie_Lou109: RT @parker_story: I can't write papers unless they're due in a matter of hours.*
Tweet-2:*RT @Dannymcfly: Happy national kiss day mwah.*
Tweet-3:*goodbye hair. one day you'll be long again.*
Tweet-4*: but you will be haha thats a lot of beer.*

The total number of retweets is three (tweet 1, tweet 2) and number of tweets is four. So the *Average retweet per tweet* = (2+1) / 4.

23) Average tweet length

First, we found the length of every tweet by counting the characters including the whitespaces. After that, we added all the lengths together to get the total length of all tweets. To find the *average tweet length* we took the total length from all tweets, divided by the total number of tweets.

For example:
Tweet-1: *but you will be haha thats a lot of beer*
Tweet-2: *goodbye hair. one day you'll be long again*
So the length of tweet-1 is 40 and tweet-2 is 43

To find *Average tweet length*= (40 + 43) / 2

24) Average time between posts

To get this feature, we used the getTimeBetweenTweets() method to find the time between tweets then we stored it in an arraylist. Then we added up all the elements in the arraylist to get *the total value,* and we divided *the total value* by the *total number of elements* in the arraylist to get the result.

For example:

Tweet Number	Time between tweet
1	0
2	6
3	8

So the *total value* is 14 because the time between post of tweet 1 and 2 is 6, and tweet 2 and 3 is 8. The *average time between posts* is 14 / 3.

25) Max idle duration between posts

We used the getTimeBetweenTweets() method to find the time between tweets then store it in an arraylist. After that, to get the maximum idle duration, we passed the arraylist to getMaxNumber() method to get the *maximum value,* which is the maximum idle duration.

For example:

Tweet Number	Time between tweet
1	0
2	6
3	8

The *max idle duration between post* is 8.

26) Average spam tweet count

This feature requires a list of spamwords. First, we checked every tweet against the list of spamwords to find out how many spamwords are in a tweet. After that, we added all the spamwords from every tweet together and divided this by the total number of tweets to get the average.

For example:
The words "silly" and "stupid" are spam words in our CSV file.
Tweet-1: *twitter is playing silly devils... I keep having to re follow people... what's going on??*
Tweet-2: *work is stupid slow. gonna be another long day*
Tweet-3: *Take a look at this video*

Thus, tweet 1 has one spamword, tweet 2 has one spamword, and tweet 3 has none; the total number of spamwords is two, and the total number of tweets is three.
The *Average spam tweet count* = 2 / 3.

27) Average URL count

For this feature, we had to find *the number of duplicated URLs*, *the number of unique URLs* and *the number of tweets* to get the average URL.

For example:
Tweet-1: *Why are so many vampire stories so weak? http://t.co/GDapgkv9*
Tweet-2: *Ask me anything http://formspring.me/discoverhannahj*
Tweet-3: *Ask me anything http://formspring.me/discoverhannahj*
Tweet-4: *WOW @smosh has done it again http://www.youtube.com/watch?v=APV5LnQvqFw*

So the *number of duplicated URLs* is two (tweet-2 and tweet-3), *the number of unique URLs* is three and *number of tweets* is four. The *Average URL count* = 2 / (3 * 4)

28) URL ratio

To find *URL ratio*, we counted the *total number of tweets contain URLs* in the content-based. We found the *Number of URLs* feature for every tweet and if a tweet had one or more than one URLs then it was considered as containing a URL. After that, we took *the total number of tweets containing URLs* and divided it by *Average URL count.*

For example:
Tweet-1: *Would you follow Loki into battle? http://t.co/s4SLqVF2, http://t.co/nLk26pqZ*
Tweet-2: *Why are so many vampire stories so weak? http://t.co/GDapgkv9*

So the number of tweets containing URLs is two, although tweet-1 has two URLs, but we do not care about the number of URLs in a tweet.
The URL ratio = 2 / *Average URL count*

29) Ratio Unique URLs

To find *ratio unique URLs,* we had to count the total number of unique URLs using HashMap array. The *ratio unique URLs* = the number of unique URL / the number of tweets.
For example:

Tweet-1: *A new week begins it's going to be awesome http://smsyo.sg*
Tweet-2: *Papal Visit - takes me back http://www.youtube.com/watch?v=aKQZwlL5x80*
Tweet-3: *Testing http://smsyo.sg*

The number of unique URLs is two (tweet-1 and tweet-3), so the ratio unique URLs = 2 / 3.

30) Average Hashtag count

This feature uses the same formula as Average URL count. However, to find the number of duplicated Hashtags, we broke a tweet into single words and used a *regular expression* to check if behind a hashtag symbol (#) contains at least one character, for example, "So bored #killmenow -_-".

31) Tweet similarity - tweet cluster

Tweet cluster is a unique tweet, so the number of tweet clusters is the number of unique tweets from a user. Instead of using cosine similarity and min distance to find tweet similarity, [11] takes the number of tweets / number of tweet clusters. To find the number of tweet clusters, we used hashmap to store it, and the number of tweet clusters was the size of the hashmap.

For example:
Tweet-1: *Love this song. http://t.co/ceb17pGU*
Tweet-2: *So happy it's spring*
Tweet-3: *Love this song. http://t.co/ceb17pGU*
Tweet-4: *A palm tree in Christmas lights*.

The number of tweet clusters is three because tweet-1 and tweet-3 are duplicated so we counted them as one cluster. The number of tweets is four, so the *tweet similarity* = 4 / 3.

32) Hashtag ratio

Same as URL ratio.

33) Total number of users mentioned

To get this feature, we extracted the *user id* in <in_reply_to_user_id> tag from every tweet, for example, <in_reply_to_user_id> 86565551 </in_reply_to_user_id> if the <in_reply_to_user_id> tag is not nil, then we counted it as one user mention.

34) getStandardDeviation(arrylistTimeBetweenTweets)

We created a getStandardDeviation() method. After that, we found the time between tweets by using the getTimeBetweenTweets() method, then added all the returned values to an arraylist and passed it to the getStandardDeviation(). This method returns a standard deviation time between tweets for a user.

35) Age of account

To find this feature, we used Twitter4j class and a function getCreatedAt() to get the date an account was created. Then we subtracted the date created from the current date.

36) Bi-directional Links

We used two functions isSourceFollowingTarget() and isSourceFollowedByTarget() from Twitter4j. Each function returned a Boolean value and to define a relationship between two users, each function had to return a "true" value.

37) Tweet similarity - Cosine Similarity

To extract this feature, we created a method called getTweetSimilarityCS() based on *Cosine Similarity* [35,[71] algorithm. Cosine Similarity between two vectors can be calculated based on the cosine of the angle between them.

38) Interaction rate

To find this feature, we took "*Total number of users mentioned*" divided by "*number of tweets*".

39) Fraction of mentions to non follower

To get this feature, we had to find out the "*number of mention to non follower*". A function called isSourceFollowedByTarget() is provided by Twitter4j API, and returns a Boolean value. If the returns value is false, it means the sender has no relationship to the receiver and in this case "number of mention to non follower" was increased by one.

The *Fraction of mention to non follower* = number of mention to non follower / number of tweets

40) maxNoofTweetRetweeted

This feature extracted information from <retweet_count> tag. After that, we added all the values from this tag to an arraylist and passed it to the getMaxNumber() method to find the maximum number of retweets.

41) minNoofTweetRetweeted

Similar to "maxNoofTweetRetweeted", but instead of using getMaxNumber() method, we used getMinNumber() method to find the minimum number of retweets.

42) medianNoofTweetRetweeted

Similar to "maxNoofTweetRetweeted", we used getMedianNumber() method to find the median number of retweets.

3.4 Hardware and Software specification

We have conducted our experiments on a PC equipped with Intel(R) Core(TM) i7-7700 CPU @ 3.60Ghz (8 CPUs), and the PC is installed with 8GB memory and 250GB SSD, which statifies our required conditions for performing all the experiments in this research. Also, we were doing all the timing based on the Java SE 8 Date and Time.

For all the testing techniques, such as ANOVA, T-Test and Equivalence testing, we will utilise the built-in functions in MS Excel. This will help us to save much time for all the calculations and also avoid all the mistakes of the calculations by hand.

To build WEST tool, we will use Netbeans platforms and some libraries, such as openCSV 4.0, and Weka package. Especially, we will need many functions provided from the Weka package to perform all the feature selection and classification processes.

3.5 Finding the best techniques to create the ASDF model

The main components of a spammer detection system are subsets of features, the number of recent tweets and classifiers. In order to find the optimisation subset of features, we extracted 172 features for every different number of recent tweets from dataset [11] such as 20, 50, 100, 150, and 200. For every number of recent tweets, we applied the feature selection algorithms: *InfoGain* and *CFSSubsetEval* to find different subsets of features and train them with five classifiers: *Naive Bayes (NB), K-nearest neighbour (IBK), Decision Tree (J48), Support Vector Machine (SMO),* and *Random Forest (RF)* to get the results for all the evaluation metrics.

First of all, we will find the optimisation number of recent tweets and the optimisation subset of features. For example, we will use ANOVA and T-Test (if required) to find the significance

difference between subsets of features (All features, Top 10, Top 20 and CFSSubsetEval) from 20 recent tweets, then pick the highest performance by their maximum and minimum interval through Equivalence Testing and repeating these steps for the rest of the number of recent tweets. So, at the end, we will have one best subset of features for each number of recent tweets.

We will use MS Excel to perform ANOVA, T-Test and Equivalence testing because this will help us to get the accurate results as our math can get messy and also it will be faster than doing all the calculating by hand. Also, in this research, we use only two-ways ANOVA without replication for all the testings.

To find the optimisation subset of features overall, we will repeat the same steps as above on the optimisation subset of features from 20RT, 50RT, 100RT, 150RT, and 200RT. Once, we found the best one overall, we chose that subset of features and the number of recent tweets and considered it as the ASDF model, because we do not know how well it is coping with the spam drift problem. After that, we will look at the classifier's performance to select the best classifier for this model based on the True Positive (TP) rate. For each of those model, we did not perform any parameter tuning on each of the classifiers. We used the classifier's default parameters settings .

3.6 Compare the ASDF model against the existing systems

Similar to finding the best techniques for the ASDF model, we repeated the same steps by using the ANOVA and T-Test (if required) to find the significant differences between the models. Then, we used Equivalence testing to find the best model based on their maximum and minimum interval.

In this research, we used TP rate, Precision, Recall, F-Measure, and Accuracy to evaluate the performance of the ASDF model and the other existing systems at predicting spammers on the testing dataset from [11]. One of the objectives in this research is finding the most efficient system; therefore, we looked at the time (feature extraction, building the model, and to classify) criteria of the ASDF to find out how long it takes to identify spammers, compared to other systems. Again, only the default parameter values are used in each of the classifiers.

3.7 Compare the ASDF model and the existing systems against Spam Drift

We will investigate how well the ASDF model and the other existing systems at handling the spam drift issue through testing this model with the dataset from [37]. Because, spammers will change their strategy or their behaviour over time to disguise themselves as a normal user, therefore testing with the latest tell us how good this model is against the spam drift problem.

We used the ANOVA and T-Test (if required) to find the significant differences between the models. Then, we used Equivalence testing to find the best model based on their maximum and minimum interval. TP rate, Precision, Recall, F-Measure, and Accuracy will be used as evaluation criteria for evaluating the performance of the ASDF model and the other existing systems against the new dataset. Just as above, we were using the default parameter settings on each of the classifiers.

3.8 Summary

This chapter describes how to implement all the current existing techniques mentioned in Chapter 2 to identify spammers in Twitter. To achieve the goals in this research, we have proposed three research questions: find the most effective features (RQ1); the most efficient model (RQ2) and the resilient model against spam drift (RQ3). After answering the research questions 1 and 2, we can create a ASDF model. To know whether it is a resilient model, which is our research question 3, we tested it against tweets from more recent dataset.We then compared ASDF performance against the existing spammer detection systems.

Also, we explained how to use the WEST tool for feature extraction, feature selection, and classification. We showed how to extract all the features from Tables 1, 2, and 3, in terms of future researchers who would like to review those features.

In the next chapter, we will show the experimental results for all the research questions and the final ASDF model. Also, we will show the experiment results for comparison between the ASDF model and the existing systems.

CHAPTER 4

RESULTS AND DISCUSSION

The previous chapters explained existing techniques to detect spammers in Twitter and how to utilise the WEST tool to implement the current techniques, such as feature extraction, feature selection, number of recent tweets and classifiers. In this chapter, we present the experimental results on the comprehensive investigative studies of the current techniques to find an efficient, effective, and resilient model to tackle the spam drift problem.

To accomplish our investigation, we have done many experiments for finding the most effective features (RQ1), and the most efficient model (RQ2) to create a ASDF model. Finally, we show the experimental results of the ASDF model tackling the spam drift problem (RQ3).

This chapter is structured as follows. In Section 4.1, we show the experiment results for finding the most effective subset of features (RQ1), and the efficient model to create the ASDF model (RQ2). Then, the comparison and the results for the ASDF model against the existing systems are discussed in Section 4.2. The best model to identify spammer against spam drift is demonstrated in Section 4.3. Finally, the summary of this chapter is presented in Section 4.4.

4.1 The experimental results for finding the ASDF

As mentioned in Section 3.4, to determine the best set of features for identifying spammers (RQ2), we performed eight different feature selection techniques: Top 1, Top 5, Top 10, Top 20, Top 50, Top 100 and CFSSubsetEval. Top-N attributes are ranked and selected based on their Information Gain value. We then compared the TP rate values produced by this subset of features against the TP rate values produced by including all the features (i.e. not applying any feature selection techniques) when they are classified with NB, SVM, KNN, DT and RF.

Many researchers have extracted features from differing numbers of recent tweets. To see whether the number of recent tweets used affects the performance of the spammer detection system, we have performed feature selection and classification of 20, 50, 100, 150 and 200 recent tweets. This section discusses the result with each of those numbers in relation to our research questions.

We performed the ANOVA technique and T-Test to find whether there is a significant difference between the results obtained by each technique. To determine which technique is the best, we used the Equivalence Testing where we compare the maximum (Average + Standard Deviation) and minimum (Average – Standard Deviation) interval for every systems.

4.1.1 - 20 Recent tweets

Table 9 shows the TP rate result for spammers between the different subset of features from the 20 recent tweets (20RT). According to the ANOVA results, there was a significant difference in the 95% confidence level between the nine subsets of features (P-value = 0.0175). Thus, we used the T-Test to find the difference between every two subsets of features. As shown in Table 48 (Appendix), "All Features" and "Top 1" was significantly different (P-value = 0.0025), while the other subsets of features were not significantly different to each other. We then used Equivalence Testing to obtain the optimisation subset of features. Based on the results, the top three subsets were: "Top 10" (AVG 60% ± 35%), "Top 20" (AVG 55% ± 26%) and "CFS-20RT" (AVG 55% ± 30%). “Top 10” produced the highest TP rate with 93% and its minimum (25%) interval is not much lower than “Top 20” (29%). Thus, we chose the "Top 10" as the optimisation subset of features,

Table 9 - Results for all subset of features in 20 recent tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-20RT
NB	**67%**	0%	**93%**	**93%**	**70%**	**80%**	**66%**	67%	42%
SMO	37%	0%	0%	0%	45%	13%	26%	42%	80%
IBK	42%	26%	22%	69%	24%	48%	41%	42%	67%
J48	51%	22%	43%	67%	43%	75%	59%	**71%**	**79%**
RF	40%	**30%**	22%	71%	33%	61%	42%	42%	**79%**

Table 10 shows the "Top 10" subsets of features for 20RT and those features were selected based on the amount of information they contributed to the class target. The numbers next to each feature indicates their information gain values. One of the "Top 10" features was *Fraction of follower per followee* that used the number of followers and followees to determine spammers in Twitter. However, [4] said we should avoid using the *Number of follower* and *Number of followee* as it could be evaded by spammers. The *Bi-Directional Link ratio* or *Interaction Rate* would be a much better indicator because spammers need to either purchase more followers and followees or have a high level of two-way interactions with the users to avoid being detected by the system. However, surprisingly, those two features were not selected in the "Top 10" subset of features for 20RT. Upon further investigation, we found that the value of the *Interaction Rate* or *Bi-Directional Link ratio* in our dataset is 0thus those features were not a good indicator for spammers.

Table 10 - Top 10 subset of features for 20 recent tweets

Top 10 subset of features for 20 recent tweets	
Max idle duration between posts (0.458)	Maximum amount of time between tweets (0.438)
Mean amount of time between tweets (0.455)	Mean word length (0.431)
Average time between posts (0.455)	Age of account (0.422)
Standard deviation tweet interval (0.45)	Fraction of follower per followee (0.422)
Tweet similarity - cosine similarity (0.44)	Average number of characters (0.395)

It can be seen that there are five time-related features in Table 10. Those features measure the user's behaviour when sending the tweets and we strongly agree with having those features in the optimisation subset of features for 20RT because, according to[1], spammers tend to have more posts than normal users over a period of time, and they cannot be idle for a long time. As shown in Figure 18, we also see this behaviour in our dataset.

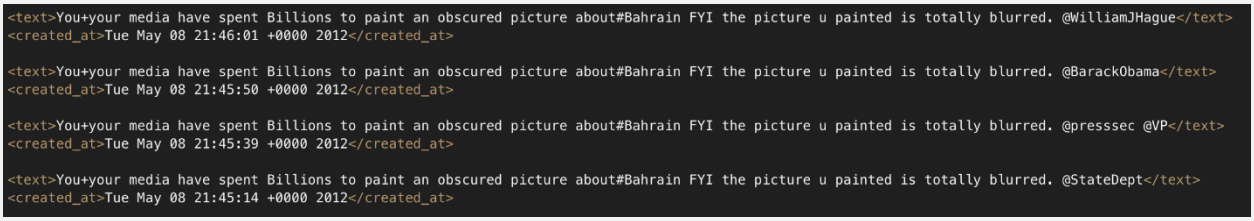


Figure 18 - Time between tweets

Table 11 shows the "CFS-20RT" subset of features for 20RT. Most of the features here could describe the spammer behaviour. For example, in our research, we found that out of 1662 spam tweets, 65 tweets contained exclamation marks and carried up to 4% of the total spam tweets, while only seven tweets contained exclamation marks in over 14,720 total ham tweets. So, this feature could help to identify spammers in Twitter. However, the time-relation features, such as *Tweets posted at 21:00 pm* and *Thirdset* (tweets posted from 6:00 am - 8:00 am) could be evaded by spammers as they could change to another posting time for spreading the tweets. [7] found that 39% of spammers had spamwords in their tweets, whereas legitimate users did not post more than 4% of their tweets containing spamwords. In our research, spammers posted 36% of spamwords in their tweets, while the legitimate users only had 12%. So it can be seen that spammers tended to post a higher percentage of spamwords than legitimate users, therefore checking the spamwords could increase the performance and better describe spammers' and legitimate users' behaviour.

Table 11 - CFSSubsetEval subset of features

CFSSubsetEval subset of features for 20RT	
Number of Exclamation marks	URL ratio
Number of spamwords	Hashtag ratio
Number of spamwords on screenname	Median number of tweets retweeted
Tweets posted at 21:00 pm	Age of account
Thirdset	Average time between posts

Although the "CFS-20RT" subset of features describes spammers behaviour well, most of the features used contextual information from the tweets to identify spammers and we believe that 20RT did not give enough information for building the spammer detection system, because 20RT is a small number of recent tweets compared to others such as 100RT and 200RT, so it did not contain much information for training and testing the system. "Top 10" subset of features focused on spammers' behaviour and did not use much contextual information, for example, and most of the features from "Top 10" were time-relation features which were extracted by calculating the time between them; however, the *URL ratio* or *Hashtag ratio* required the contextual information to extract. Therefore, the "Top 10" subset of features performed better than the "CFS-20RT" in 20RT.

4.1.2 - 50 Recent tweets

Table 12 shows the TP rate result for spammers between the different subsets of features from the 50 recent tweets (50RT). According to the ANOVA results, there was no significant difference in the 95% confidence level between the nine subsets of features (P-value = 0.0511). The results of the Equivalence Testing showed that "Top 10" (AVG 56% ± 33%), "CFS-50RT" (AVG 56% ± 27%) and "Top 20" (AVG 49% ± 21%) were the top three subsets of features in 50RT. Although it appears that "CFS-50RT" has a lower variance than the "Top 10", "Top 10" produced lower variance than "CFS-50RT" if we removed the results from SMO and "Top 10" achieved 92% with NB, that was the highest result throughout the classifiers from the other subsets of features. Thus, "Top 10" is the optimisation subset of features in 50RT.

Table 12 - Results for all subset of features in 50 recent tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-50RT
NB	**66%**	0%	**97%**	**92%**	**66%**	66%	**58%**	**75%**	30%
SMO	44%	0%	0%	0%	50%	15%	37%	45%	22%
IBK	24%	22%	20%	62%	24%	42%	34%	38%	68%
J48	43%	**26%**	20%	61%	43%	**67%**	52%	48%	**80%**
RF	32%	22%	11%	65%	33%	57%	41%	34%	**80%**

In Table 13, most of the features selected in "Top 10" defined the spammers behaviour well, except for the *Fraction of follower per followee* because this feature could be evaded by spammers, as they could purchase more followers and followees.

Table 13 - Top 10 subset of features for 50 recent tweets

Top 10 subset of features for 50 recent tweets	
Average time between posts (0.482)	Maximum number of time between tweets (0.466)
Mean amount of time between tweets (0.481)	Mean word length (0.461)
Max idle duration between posts (0.48)	Fraction of follower per followee (0.449)
Standard deviation tweet interval (0.475)	Age of account (0.44)
Tweet similarity - cosine similarity (0.473)	Average number of characters (0.44)

Table 14 shows the "CFS-50RT" of features extracted from 50RT. Most of the features in Table 14 were the same as "CFS-20RT" from 20RT, except 50RT did not contain the *Average number of time between tweets, tweets posted at 21:00 pm*, and *Third Set.* According to the feature characteristics, the "CFS-20RT" features from 20RT were more effective at identifying spammers than 50RT, because *Is Social Media* was not an effective feature because it is time-consuming to extract and did not describe well spammer behaviors. For 50RT, 33% of tweets posted by spammers used @mention and only 31% of normal users, so the *Number of Mentions* was not an effective feature to identify spammers because spammers and normal users used almost the same number of @mention in their tweets, therefore it was hard to determine spammers based on this feature.

Table 14 - CFSSubsetEval subset of features for 50 recent tweets

CFSSubsetEval subset of features for 50RT	
Number of Mentions	Number of spamwords on screen name
Number of Exclamation marks	Maximum amount of time between tweets
Number of Spam words	URL ratio
Is Social Media	Hashtag ratio
Median Number Of tweet retweeted	Age of account

Back to Table 12, the maximum TP rate for CFSSubsetEval subset of features was 80%, based on J48 and RF classifiers, so the performance of this subset of features was lower than the "Top 10". It can be seen that most of the features from the "Top 10" were more focused on spammer behaviours than "CFS-50RT". So this subset of features achieved a lower performance than the "Top 10", because the model could not satisfactorily capture spammer behaviour.

4.1.3 - 100 Recent tweets

Table 15 shows the TP rate results for spammers between the different subsets of features from 100 recent tweets (100RT). According to the ANOVA results, there was a significant difference in the 95% confidence level between the nine subsets of features (P-value = 0.0005). Therefore, we performed the T-Test to find out the difference between every two subsets of features. As shown in Appendix - 100 Recent tweets, only "All Features and Top 1" (P-value = 0.0133) and "All Features and Top 5" (P-value = 0.0210) were significantly different, while the other subsets of features were not significant to each other. We then used the Equivalence Test to obtain the optimisation subset of features and the top three subsets of features were: "Top 10" (AVG 76% ± 43%), "CFS-Top100" (AVG 75% ± 33%) and "Top 20" (AVG 74% ± 37%).

Table 15 - Results for all subset of features in 100 recent tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-100RT
NB	61%	0%	11%	81%	39%	53%	**55%**	66%	48%
SMO	**69%**	0%	0%	0%	14%	17%	49%	**68%**	31%
IBK	12%	15%	19%	**100%**	33%	**100%**	25%	17%	**100%**
J48	50%	**24%**	**26%**	**100%**	69%	**100%**	50%	52%	**100%**
RF	23%	15%	17%	**100%**	**71%**	**100%**	35%	30%	**100%**

By looking at Table 15, it can be seen that “Top 10”, “Top 20”, and “CFS-100RT” achieved 100% TP rate based on IBK, J48 and RF classifiers, while the rest of the subset of features did not achieved 100% TP rate on their classifiers. But, the minimum interval of the “CFS-100RT” (42%) is the highest compare to the “Top 10”, and “Top 20”. Because, we would like to choose a model that could detect the most spammers possible, therefore, we chose “CFS-100RT” to be the optimisation subset of features in 100RT.

Table 16 - CfsSubsetEval subset of features for 100 recent tweets

CFSSubsetEval subset of features for 100RT	
Number of mentions	Fraction of follower per followee
Number of exclamation marks	URL ratio
Number of spamwords	Median number of tweet retweeted
Number of spamwords in screenname	Age of accounts

Table 16 shows the subset of features selected by CFSSubsetEval algorithm for 100RT and this subset of features was similar to the "CFS-20RT" and "CFS-50RT", except it did not contain the *Is Social Media* and *Third Set* and, as mentioned above, these features did not help to increase the performance. However, this subset of features performed very well in 100RT and achieved 100% TP rate by IBK, J48, and RF classifiers. However, there is one concern about this subset of feature that is *Number of spamwords* feature. It is a good feature to build a spammer detection system because spammers tended to post a higher percentage of spamwords than legitimate users, but this feature requires time to extract, and spammers can invent new spamwords, so we needed to update our spamword list, regularly, to make sure the system can capture all the spamwords.

4.1.4 - 150 Recent tweets

Table 17 shows the TP rate results for spammers between the different subset of features from 150 recent tweets (150RT). According to the ANOVA results, there was no significant difference in the 95% confidence level between the nine subsets of features (P-value = 0.3443). So, we did not need to perform the T-Test to find the difference between every two subsets of features. We calculated and compared the maximum and minimum TP rate interval of every subset of features and found the top three subsets were: ""CFS-150RT" (AVG 52% ± 21%), "Top 1" (AVG 40% ± 46%) and "Top 15" (AVG 48% ± 21%). It surprised us when increasing the number of recent tweets to 150 RT that "Top 1", "Top 15" and "CFS-150RT" were in the top three subsets of features. "Top 1" and "CFS-150RT" outperformed "Top 15" based on the maximum interval, but the minimum interval of "Top 1" was 0% compared to 26% and 30% of "Top 5" and "CFS-150RT", respectively. "Top 1" achieved 90% of the TP rate from IBK and RF classifiers, but we chose "CFS-150RT" instead of "Top 1" because it is the second highest maximum interval and the highest minimum interval. Also, if we built a model with only one feature, spammers would evade that feature easily. Therefore, we did not choose "Top 1" to be the optimisation subset of features for 150RT.

Table 17 - Results for all subset of features in 150 recent tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-150RT
NB	**63%**	0%	19%	**61%**	42%	**67%**	**62%**	**61%**	42%
SMO	33%	0%	0%	0%	14%	14%	12%	25%	22%
IBK	15%	**90%**	14%	13%	**66%**	48%	15%	15%	73%
J48	34%	20%	**44%**	37%	56%	49%	57%	52%	52%
RF	34%	**90%**	15%	40%	62%	57%	46%	30%	**74%**

Table 18 - CfsSubset subset of features for 150 recent tweets

CFSSubset subset of feature for 150 recent tweets	
Number of mentions per word	Fraction of follower per followee
Number of exclamation marks	URL ratio
Number of spam words	Median number of tweet retweeted
Is Social Media	Age of account
Number of spam words on screen name	

Table 18 shows all the selected features based on the CFSSubsetEval algorithm, and Table 17 shows that the highest TP rate was 74%, which was not very efficient in detecting spammers compared to the other subset of features from 20RT, 50RT, and 100RT. The factors that affected this result were that some of the features were not very effective in describing the spammer characteristics. As mentioned in Section 4.1.1, Fraction of follower and followee is not good because spammers could evade *Number of followers* and *followees* by purchasing followers and followee online, and *Is Social media* is not good because it is time-consuming to extract these features, and also we need to update the list of social media regularly to avoid missing new ones.

4.1.5 - 200 Recent tweets

Table 19 shows the TP rate results for spammers between the different subset of features from the 200 recent tweets (200RT). According to the ANOVA results, there was a significant difference in the 95% confidence level between the nine subsets of futures (P-value = 1.2315E-05). Thus we used the T-Test to find out the difference between every two subsets of features. As shown in Appendix - 200 Recent tweets, "All Features and Top 1" (P-value = 0.0139), "All Features and Top 5" (P-value = 0.0049), and "All Features and Top 10" (P-value = 0.0325) were significantly different, while the other subsets of features were not significantly different to each other. The Equivalence Testing results showed that "CFS-200RT" (54% ± SD: 22%), "Top 15" (44% ± SD: 19%) and "Top 20" (45% ± SD: 23%), and were the Top 3 subsets of features. Out of the three best subsets of features in 200RT, we chose "CFS-200RT" as the optimisation subset of features overall, because the IBK classifier achieved a 79% TP rate, which was the highest result compared to "Top 15" and "Top 20" subset of features. Also, the overall performance of "CFS-200RT" was better than "Top 10" and "Top 20" based on the equivalence testing results as “CFS-200RT” achieved the highest maximum and minimum intervals.

Table 19 - Results for all subset of features in 200 recent tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-200RT
NB	**60%**	0%	13%	23%	39%	**72%**	**58%**	**55%**	41%
SMO	33%	0%	0%	0%	13%	13%	37%	32%	23%
IBK	16%	**21%**	10%	21%	53%	29%	34%	20%	**79%**
J48	54%	**21%**	**15%**	**26%**	56%	56%	52%	52%	58%
RF	28%	**21%**	10%	22%	**63%**	57%	41%	35%	71%

Table 20 shows the subset of features selected by CFSSubsetEval algorithm from 200RT. Similar to 150RT, this subset of features did not optimise the performance for 200RT. Same as 150RT, we did not expect the *Fraction of follower per followee, Is Social Media* and *URL ratio* features to produce an effective model for spammer detection systems, as those features did not describe spammer characteristics well.

Table 20 - CfsSubsetEval subset of features for 200 recent tweets

CfsSubset subset of features for 200 recent tweets	
Number of exclamation marks	Fraction of follower per followee
Number of spam words	URL ratio
Is Social Media	Median number of tweets retweeted
Number of spam words in screen name	Age of account

4.1.6 The most effective model through feature selection

This section was used to find the most effective content-based and user-based features (RQ1). Table 21 shows the combination of the optimisation subset of features from different numbers of recent tweets. According to the ANOVA results, there was no significant difference in the 95% confidence level between the five subsets of features (P-value = 0.2741), so it was not necessary to perform a T-Test.

Table 21 - A combination of best subset from different numbers of recent tweets

	Top10-20RT	**Top10-50RT**	**CFS-100RT**	**CFS-150RT**	**CFS-200RT**
NB	**93%**	**92%**	48%	42%	41%
SMO	0%	0%	31%	22%	23%
IBK	69%	62%	**100%**	73%	**79%**
J48	67%	61%	**100%**	52%	58%
RF	71%	65%	**100%**	**74%**	71%

Based on the Equivalence Testing results in Table 22, "CFS-100RT" had the highest maximum and minimum intervals compare to the other four subset of features. Thus, we chose "CFS-100RT" as the optimisation subset of features and the optimisation number of recent tweets overall.

Table 22 - Equivalence test results of the optimisation subset of feature

	Top10-20RT	**Top10-50RT**	**CFS-100RT**	**CFS-150RT**	**CFS-200RT**
Max	95%	89%	**109%**	74%	77%
Min	24%	22%	**42%**	30%	31%

Table 23 shows the most effective features found in this research and they were extracted from 100RT through the CFSSubsetEval algorithm.

Table 23 - The most effective features to identify spammers in twitter

The most effective features extracted from CFS-100RT	
Number of mentions	Fraction of follower per followee
Number of exclamation marks	URL ratio
Number of spamwords	Median number of tweet retweeted
Number of spamwords in screenname	Age of accounts

4.1.7 The most efficient model through feature selection

This section answers a part of research question two: "***Which model is the most efficient at identifying spammers in Twitter?".*** In the previous section, the number of recent tweets is one of the factors that might affect making an efficient model, and another factor that could affect the model is the classifier.

Table 24 shows in minutes the total time for feature extraction, time to build a model and to classify. We can see that there are two group of features: the first group is "Top10-20RT and Top10-50RT" and this group used Information Gain algorithm for feature selection process. The second group is "CFS-100RT, CFS-150RT, and CFS-200RT" and this group used CFSSubsetEval algorithm to find the relevant features.

"Top10-20RT" used less time to build the model than "Top10-50RT" because "Top10-20RT" used less number of tweets than "Top10-50RT", thus, it required less time for feature extraction and classification processes. Similarly, with "CFS-100RT", "CFS-150RT", and "CFS-200RT", "CFS-100RT" used less time than the other two subset of features because it used a smaller number of tweets than the other two subset of features. Furthermore, "CFS-150RT" and "CFS-200RT" extracted the *Is Social Media* feature from the tweets which requires the system to go through a CSV file that stores all the social media domain and check them against every tweets. Thus, "CFS-150RT" and "CFS-200RT" are slower to build than "CFS-100RT".

Table 24 -Time to build the systems in minutes

	Top10-20RT	**Top10-50RT**	**CFS-100RT**	**CFS-150RT**	**CFS-200RT**
NB	3	7	2	3	5
SMO	2	6	3	4	6
IBK	4	9	11	13	17
J48	5	9	4	5	7
RF	3	7	**3**	3	5

To find the efficient model, we are relying on the equivalence testing results in Table 22, and according to the results, the "CFS-100RT" subset of features is the best performance overall. In Table 24, we can see that the top two fastest model are"Top10-20RT" and "CFS-100RT". Although it is faster to build "Top10-20RT" model than "CFS-100RT", its TP rate is considerably lower than "CFS-100RT." Therefore, we are considering "CFS-100RT" as the most efficient model and RF is the chosen classifier for this model because it is achieved 100% TP rate, instead

of NB achieved only 48% TP rate. Furthermore, RF is slower than NB by 1 minute only but the RF result is much higher than NB. Hence, we are considering RF is the best classifier.

4.2 The ASDF vs the other existing systems

In this section, we will compare the performance of our ASDF model with the existing systems in terms of TP rate, Precision, Recall, F-measure, Accuracy and Time.

TP Rate

Table 25 shows the TP rate between our ASDF model and the existing systems. It can be seen that, out of five classifiers, the ASDF model achieved 100% in IBK, J48 and RF classifiers. We were performing the ANOVA technique to determine if there is any significant difference between the systems. Based on the ANOVA results, there is a significant difference at 95% confidence level between the systems (P-value = 3.8643E-05).

Thus, we carried out the T-Test technique to find out the significant difference between our ASDF model against the existing systems and the T-Test results showed that there is significant difference between "ASDF vs [6] (P-value = 0.02)", "ASDF vs [1] (P-value = 0.04)", "ASDF vs [14] (P-value = 0.004)", "ASDF vs [43] (P-value = 0.02)" and "ASDF vs [11] (P-value = 0.01)".
While, there are no significant difference between ASDF model and the rest of the systems. Then, we performed the equivalence testing to compare the maximum and minimum intervals of the systems to find out the best technique in term of TP rate.

Table 25 - TP rate between ASDF model and existing systems

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
NB	80%	30%	0%	10%	60%	30%	50%	40%	30%	40%	30%
SMO	0%	10%	0%	0%	40%	10%	20%	0%	10%	0%	10%
IBK	**100%**	30%	50%	10%	30%	40%	40%	50%	40%	20%	30%
J48	**100%**	50%	50%	20%	70%	40%	60%	80%	60%	30%	50%
RF	**100%**	40%	60%	0%	30%	40%	40%	70%	50%	10%	70%

Table 26 shows all the equivalence testing results for ASDF model and the existing systems based on TP rate. The ASDF model gained the highest maximum and minimum interval compared to the rest of the systems, thus, we considered the ASDF model is the best performance in term of TP rate.

Table 26 - Maximum and minimum intervals of the systems based on TP rate

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
Max	**120%**	50%	60%	20%	60%	50%	60%	80%	60%	40%	60%
Min	**30%**	20%	0%	0%	30%	20%	30%	20%	20%	0%	20%

Precision

Table 27 shows the precision between ASDF and the existing systems. It can be seen that, the highest precision in this table is 100% and the lowest is 0%, which mean that classifiers cannot predict any spammers. ASDF and [44] achieved 100% precision but ASDF can achieve them using IBK, J48, and RF, while [44] based on SMO classifier only. Hence, it seems like the ASDF's features are better than [44]. ANOVA results showed that there is a significant difference between the ASDF against the existing systems at 95% confidence level (P-value = 0.007). Therefore, we also performed T-Test technique to find the significant difference between ASDF against every single system. According to the T-Test results, only two systems are significantly different to ASDF: "ASDF vs [14] (P-value = 0.02)" and "ASDF vs [11] (P-value = 0.03)".

Table 27 - Precision of ASDF vs the existing systems

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
NB	17%	30%	0%	10%	37%	68%	38%	25%	28%	10%	28%
SMO	0%	72%	0%	0%	38%	80%	73%	0%	100%	0%	37%
IBK	**100%**	36%	56%	13%	53%	43%	50%	64%	44%	22%	69%
J48	**100%**	36%	20%	13%	56%	39%	49%	77%	56%	28%	53%
RF	**100%**	71%	67%	80%	65%	61%	76%	88%	69%	23%	81%

We have performed equivalence testing to find the best model based on its performance. According to the Table 28, the ASDF model has the highest maximum interval, while the minimum number of interval is lower than [6], [7], [43], [15], [40], [44] and [10]. However, in Table 27, our model achieved 100% precision with three classifiers, which is considerably higher than the maximum interval of all the other systems, thus ASDF model outperformed the others in term of precision.

Table 28 - Equivalence testing results of Precision for the systems

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
Max	**113%**	69%	75%	14%	61%	75%	73%	87%	86%	27%	75%
Min	**12%**	28%	3%	3%	37%	41%	40%	13%	32%	5%	31%

Recall

Table 29 shows that the ASDF model achieved the best recall performance compare to the existing systems. It achieved 100% with IBK, J48, and RF, and 81% with NB classifier, which is still the higher than the recall performance of the other systems. To determine the significant difference between the systems, we carried out the ANOVA technique and based on the results there is a significant difference at 95% confidence level between the systems (P-value = 3.8643E-05). According to the T-Test results, there are five systems that are significanty different to ASDF: "ASDF vs [6] (P-value = 0.02)", "ASDF vs [1] (P-value = 0.04)", "ASDF vs [14] (P-value = 0.004)", "ASDF vs [43] (P-value = 0.02)", and "ASDF vs [11] (P-value = 0.14)".

Table 29 - Recall of ASDF model and the existing systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	81%	25%	0%	11%	62%	30%	54%	36%	35%	43%	34%
SMO	0%	14%	0%	0%	38%	10%	22%	0%	13%	0%	13%
IBK	**100%**	32%	54%	14%	27%	42%	36%	55%	41%	22%	34%
J48	**100%**	46%	46%	17%	65%	36%	61%	76%	57%	27%	46%
RF	**100%**	40%	59%	4%	34%	39%	35%	73%	45%	14%	66%

Table 30 shows the maximum and minimum interval values for all the systems and it can be seen that ASDF model gained the highest maximum and minimum interval. Thus, we considered ASDF is the best one in term of recall.

Table 30 - Maximum and Minimum interval of recall for the systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	**119%**	43%	61%	16%	62%	44%	57%	79%	54%	37%	57%
Min	**32%**	18%	2%	2%	28%	18%	25%	16%	21%	5%	19%

F-measure

Table 31 shows the F-measure results for all the systems, and similar to Precision and Recall, our ASDF model achieved 100% on IBK, J48 and RF as well. On the other hand, [14] is the worst performing system because it obtained only 15% F-Measure. In this evaluation metric, the ANOVA results showed that there is significant difference at 95% confidence level (P-value = 0.0001). We also carried out the T-Test technique and the results showed that there are two systems that are significantly different to ASDF: "ASDF vs [14] (P-value = 0.01)", and "ASDF vs [11] (P-value = 0.02)".

Table 31 - F-Measure results for all the systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	28%	28%	0%	10%	46%	41%	45%	30%	31%	16%	31%
SMO	0%	24%	0%	0%	38%	18%	33%	0%	22%	0%	19%
IBK	**100%**	34%	55%	14%	35%	43%	42%	59%	43%	22%	46%
J48	**100%**	41%	57%	15%	60%	37%	55%	77%	56%	27%	49%
RF	**100%**	51%	63%	6%	45%	47%	48%	80%	55%	17%	73%

According to Table 32, our maximum interval is 113% which is the highest compare to the rest of the systems, and [7] gained the highest minimum interval, which is 35%. However, [7] maximum interval (54%) is much lower than ASDF, thus we considered ASDF model to be the best model

in terms of F-measure. Meanwhile, [14] is the worst model as its maximum and minimum interval are the lowest.

Table 32 - Equivalence testing results for F-measure of the systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	**113%**	46%	67%	15%	54%	48%	52%	83%	56%	26%	63%
Min	**17%**	24%	2%	2%	35%	25%	36%	15%	26%	6%	23%

Accuracy

Table 33 shows the accuracy results of the systems. Our ASDF model achieved 100% based on IBK and RF and it is the most accurate model. However, with the NB classifier, ASDF achieved only 51% which is the lowest comparing to the other systems. According to the ANOVA results there is no significant difference at 95% confidence level between the systems (P-value = 0.53). Thus, we do not need to perform T-Test here.

To determine how well all the systems performing and also to find out the best one, we have calculate the maximum and minimum interval values for each of the systems as shown in Table 34.

Table 33 - Accuracy results of the systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	51%	90%	89%	84%	88%	91%	89%	88%	89%	65%	83%
SMO	88%	93%	89%	92%	90%	91%	93%	93%	93%	91%	88%
IBK	**100%**	90%	90%	85%	92%	89%	92%	94%	92%	87%	91%
J48	**99%**	90%	92%	84%	93%	88%	91%	96%	93%	88%	90%
RF	**100%**	94%	92%	88%	93%	91%	93%	97%	94%	89%	94%

According to the results shown in Table 34, it can be seen that the ASDF model gained the highest maximum (100%) interval. However, we chose [40] to be the best system in terms of accuracy because its maximum interval is 97% and its minimum interval is 90%. [40] maximum interval is about 100%, while its minimum interval is the highest compare to the other systems.

Table 34 - Equivalence testing results for accuracy of the systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	100%	93%	91%	90%	93%	91%	93%	**97%**	94%	94%	93%
Min	66%	89%	88%	83%	89%	88%	89%	**90%**	90%	73%	85%

Time

Table 35 shows the total time required to extract features, to build the model and to classify every system in minutes. In this research ASDF model is the fastest model as it took only 3 minutes to build and classify. Also, it can be seen that [10] took 15 minutes, and it is the second fastest system in this research. However, there is one feature that is not effective in identifying spammers such as *Fraction Of Mention Contain URL* and, according to our dataset, we found a very small number of @mention tweets containing the URL, which means the *Fraction Of Mention Contain URL* did not contribute too much to identifying spammers.

Additionally, [14] system is the longest system in this research. Most of the features used in this system used the CSV file to check the tweets. For example, to check whether or not a tweet contained a city name needed a CSV file that contained many cities. Thus, the time it took to extract the feature depends on the size of the CSV file and the number of tweets. Also, [14] performance was not very impressive compared to other existing systems. Hence, the [14] subset of features was not very effective and efficient in identifying spammers.

ASDF is the fastest model in this research because the "CFS-100RT" subset of features are easy to extract as those features did not require any complicated steps to extract like *Maximum number of tweets per week* or *Minimum number of time between tweets*. Those features only take information from the tweets and most of them do not require any CSV files for feature extraction. The only two features require CSV files in "CFS-100RT" are the *Number of spam words* and *Number of spam words in screenname*. However, these two features are good to describe spammers, because spammers contain more spam words in their tweets than legitimate users.

Table 35 - Time recorded for every system in minutes

The systems	Time (minutes)
ASDF	**3**
[10]	15
[1]	20
[44]	27
[6]	30
[15]	35
[40]	112
[43]	158
[7]	565
[11]	793
[14]	897

4.3 Best model to tackle with spam drift problem

In previous sections, we used WEST to performed all the experiments and we found the "CFS-100RT" is the optimisation subset of features to identify spammers in Twitter. In this experiment,

we also test this subset of features against spam drift problem (i.e. on a second test dataset containing tweets taken from period newer than the training dataset) and according to the results shown in Table 36, it is detecting 91% TP rate based on the new dataset [39] which has tweets taken from 3 years after the tweets in the training dataset. Hence, its performance decreased 9% compared to its performance on our first test dataset [11].

According to [21], spammers were using 3rd party API to post more tweets throughout the day. [72] also found that spammers post high numbers of tweets (more than 50 tweets per day). In the new dataset, most of the spammers posted a very small number of tweets in one day as shown in Figure 19; the number of users who posted less than 10 tweets is more than 3000 users, while there are a very small number of users who post more than 10 tweets per day. That means in the new dataset, the spammer behaviour has changed; they posted fewer tweets to disguise themselves as legitimate users. However, as shown in Table 36, the ASDF model is still coping well against spam drift problem as it obtained 91% TP rate.

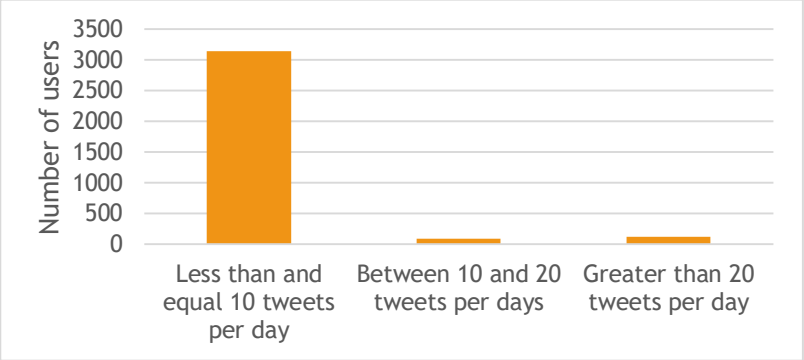


Figure 19 - Number of tweets posted by users

Although spammers were changing their behavior by modified the number of tweets posted in one day, ASDF was not affected because "CFS-100RT" subset of features does not contain any time-relation features. “CFS-100RT” subset of features focused more on spammer characteristics for example, *Number of mentions, Number of spam word, Number of Spamword on screen name,* or *URL ratio*. Spammers cannot spread their contents to the users without using @mention or without including URL in their tweets, therefore, "CFS-100RT" subset of features could detect spammers even when there are the spam drift phenomena. However, WEST is still necessary and a novel tool for researchers to test and evaluate their spammer detection systems, because the users could quickly test their subset of features against spam drift or modify their subset of features if it is not coping well with spam drift problem.

We also tested the performance of the existing systems against spam drift problem (i.e. we tested the existing system on the new dataset [39]) and compared it to ASDF in terms of TP rate, Precision, Recall, F-measure, and Accuracy.

TP rate

Table 36 below showed that the TP rate of ASDF model is the best. Although [6]'s system achieved 100% with IBK and is the highest results, it is not a good set of features because its performance is not consistent. With [6] we can only detect spammers when we use IBK classifiers. Meanwhile, ASDF could detect spammers with at least 50% TP rate regardless of the classifiers.

We have performed the ANOVA techniques for finding the significant difference between the systems and based on the results, there is significant difference at 95% confidence interval between the systems (P-value = 2.3359E-08). We also performed the T-Test techniques between ASDF and against every systems and the results showed that the ASDF model has significant difference with the existing systems: ("ASDF vs [6] (P-value = 0.02)", "ASDF vs [1] (P-value = 0.0001)", "ASDF vs [14] (P-value = 0.0005)", "ASDF vs [7] (P-value = 0.0001)", "ASDF vs [43] (P-value = 0.0001)", "ASDF vs [15] (P-value = 0.0001)", "ASDF vs [40] (P-value = 0.0001)", "ASDF vs [44] (P-value = 0.0001)", "ASDF vs [11] (P-value = 0.0005)", "ASDF vs [10] (P-value = 0.0002)".

Table 36 - Performance results of the ASDF model vs existing systems based on TP rate

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	80%	0%	0%	30%	0%	0%	0%	0%	0%	30%	20%
SMO	80%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
IBK	75%	**100%**	0%	0%	0%	0%	0%	0%	0%	0%	0%
J48	50%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
RF	91%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

To determine the optimisation subset of features, we have calculated the maximum and minimum intervals for every systems as shown in the Table 37 below. It can be seen that the ASDF model achieved the highest maximum and minimum interval against the existing systems. This means that the existing systems cannot cope well with spam drift problem as they all achieved very low maximum and minimum interval.

Table 37 - Maximum and minimum interval of TP rate of the systems against spam drift problem

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	90%	60%	0%	2%	0%	0%	0%	0%	0%	2%	1%
Min	60%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Precision

Table 38 showed the experiment results between ASDF and the existing systems based on Precision. [10] obtained 100% precision with NB, SMO and IBK classifiers but got 0% with J48 and RF. [6] and ASDF achieved 80% of precision, which is the second highest result. However, the ASDF model is more consistent the the rest of the classifiers because this model is able to detect spammers regardless of the classifiers.

We have performed the ANOVA technique to see the significant difference between ASDF model and the existing systems and based on the results, there is a significant difference between the

systems (P-value = 5.4757E-07). Therefore, we also carried out the T-Test technique and the results showed that the ASDF model is signicantly different to all of the existing systems: ("ASDF vs [6] (P-value = 0.002)", "ASDF vs [1] (P-value = 9.0301E-05)", "ASDF vs [14] (P-value = 0.003)", "ASDF vs [7] (P-value = 9.0301E-05)", "ASDF vs [43] (P-value = 9.0301E-05)", "ASDF vs [15] (P-value = 2.2601E-05)", "ASDF vs [40] (P-value = 9.0301E-05)", "ASDF vs [44] (P-value = 9.0301E-05)", "ASDF vs [11] (P-value = 0.003)", "ASDF vs [10] (P-value = 0.24)".

Table 38 - Precision results of ASDF model and existing systems against spam drift

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	70%	0%	0%	50%	0%	0%	0%	0%	0%	50%	**100%**
SMO	70%	0%	0%	0%	0%	0%	0%	0%	0%	0%	**100%**
IBK	**100%**	80%	0%	0%	0%	0%	10%	0%	0%	0%	**100%**
J48	80%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
RF	70%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

To determine the best system in term of Precision, we calculated the maximum and minimum intervals for the systems and Table 39 showed that [10] achieved the maximum interval with 110% which is the highest result, while ASDF achieved only 90% of maximum interval which is the second highest result. However, the minimum interval of [10] is much lower comparing to ASDF model. Because the maximum interval of ASDF is not very much lower than [10], we are considering ASDF model as the best technique in term of Precision.

Table 39 - Equivalence testing results of Precision for the systems against spam drift

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	90%	50%	0%	3%	0%	0%	1%	0%	0%	3%	110%
Min	60%	0%	0%	0%	0%	0%	0%	0%	0%	0%	10%

Recall

Table 40 showed the results of recall for the systems against spam drift phenomena. It can be seen that, although [6] obtained the highest recall (100%) with IBK, this system cannot detect any spammers if they use other classifiers such as NB, SMO, J48 and RF. As shown in Table 40, the ASDF model got the second highest recall values of 90% with RF and it can detects at least 50% spammers in term of spam drift with any classifiers.

Table 40 - Results of recall of ASDF and the existing systems against spam drift

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	80%	0	0	30%	0	0	0	0	0	30%	20%
SMO	80%	0	0	0	0	0	0	0	0	0	0
IBK	80%	**100%**	0	0	0	0	0	0	0	0	0
J48	50%	0	0	0	0	0	0	0	0	0	0
RF	90%	0	0	0	0	0	0	0	0	0	0

We have performed the ANOVA techniques for the systems in Table 40 above and it shown that there is a significant difference at 95% confidence level between the systems (P-value = 1.5586E-08). Thus, we carried out the T-Test for the systems and according to the results the ASDF model is significantly different to the existing systems: ("ASDF vs [6] (P-value = 0.02)", "ASDF vs [1] (P-value = 0.0001)", "ASDF vs [14] (P-value = 0.0005)", "ASDF vs [7] (P-value = 0.0001)", "ASDF vs [43] (P-value = 0.0001)", "ASDF vs [15] (P-value = 0.0001)", "ASDF vs [40] (P-value = 0.0001)", "ASDF vs [44] (P-value = 0.0001)", "ASDF vs [11] (P-value = 0.0005)", "ASDF vs [10] (P-value = 0.0003)". We also calculated the maximum and minimum intervals for every systems as shown in Table 41 below. It can be seen that, ASDF model is achieving the highest maximum and minimum intervals. So, in term of recall, we consider ASDF as the best model.

Table 41 - Maximum and Minimum intervals of the systems against spam drift in term of Recall

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
Max	90%	60%	0%	20%	0%	0%	0%	0%	0%	20%	10%
Min	60%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

F-measure

According to Table 42, most of the existing spammer detections systems did not performing well at handling the spam drift problem. In this evaluation metrics, the ASDF model and [6] performed well at detecting spammers. However, [6] only achieved 90% based on IBK classifier, which is the highest result. But in term of consistency, the ASDF model achieved 60% based on J48 and 80% with the other classifiers. So, it seems like the ASDF model is outperforming the other existing systems.

We carried out the ANOVA techniques and the results showed that there is a significant difference between the systems at 95% of confidence interval (P-value = 1.2207E-08). Also, in the T-Test results, the ASDF model is significantly different to the rest of the systems as well ("ASDF vs [6] (P-value = 0.01)", "ASDF vs [1] (P-value = 2.2601E-05)", "ASDF vs [14] (P-value = 0.0005)", "ASDF vs [7] (P-value = 2.2601E-05)", "ASDF vs [43] (P-value = 2.2601E-05)", "ASDF vs [15] (P-value = 2.2601E-05)", "ASDF vs [40] (P-value =2.2601E-05)", "ASDF vs [44] (P-value = 2.2601E-05)", "ASDF vs [11] (P-value = 0.0005)", "ASDF vs [10] (P-value = 0.0005)".

Table 42 - F-Measure results for ASDF and the existing systems against spam drift

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
NB	80%	0	0	40%	0	0	0	0	0	40%	40%
SMO	80%	0	0	0	0	0	0	0	0	0	0
IBK	80%	**90%**	0	0	0	0	0	0	0	0	0
J48	60%	0	0	0	0	0	0	0	0	0	0
RF	80%	0	0	0	0	0	0	0	0	0	0

We calculated the maximum and minimum interval for the systems to determine the best one as shown in Table 43. Based on the results, we can see that ASDF model is outperforming the existing systems because its maximum and minimum intervals are the highest. Therefore, we are considering ASDF is the best model in term of F-Measure.

Table 43 - Maximum and Minimum intervals of the systems against spam drift in term of F-Measure

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	80%	60%	0%	30%	0%	0%	1%	0%	0%	30%	30%
Min	70%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Accuracy

Table 44 showed the Accuracy of the systems and it can be clearly seen that all the systems could detect spammers with 2% to 81% accuracy, which is much better then the previous evaluation metrics. However, similarl to the other four evaluation metrics, the ASDF model and [6] are the highest performance systems compared to the rest of the existing systems.

For finding the best model in Accuracy, we performed the ANOVA technique and the results showed that there is a significant difference at 95% confidence interval on the systems (P-value = 1.1865E-07). In term of Accuracy, the T-Test showed that there is no significant difference between "ASDF vs [6] (P-value = 0.07) ". However, there is a significant difference between ASDF model and the rest of the existing systems "ASDF vs [1] (P-value = 0.0013)", "ASDF vs [14] (P-value = 0.0013)", "ASDF vs [7] (P-value = 0.0014)", "ASDF vs [43] (P-value = 0.0011)", "ASDF vs [15] (P-value = 0.0006)", "ASDF vs [40] (P-value = 0.0013)", "ASDF vs [44] (P-value = 0.0010)", "ASDF vs [11] (P-value = 0.0014)", "ASDF vs [10] (P-value = 0.0007)".

Table 44- Accuracy results of ASDF and the existing systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	69%	38%	40%	42%	38%	40%	26%	40%	40%	42%	43%
SMO	61%	40%	40%	40%	40%	40%	27%	40%	40%	39%	28%
IBK	62%	**81%**	40%	40%	40%	27%	2%	40%	30%	40%	28%
J48	58%	40%	40%	40%	40%	40%	27%	40%	40%	40%	27%
RF	80%	40%	40%	40%	40%	40%	27%	40%	40%	39%	27%

Table 45 below showed the maximum and minimum intervals of the systems. Based on the results, the top two systems are: the ASDF model and [6]. [6] achieved the highest maximum intervals (70%) but for the minimum interval; the ASDF model is the highest (60%). Thus, we consider ASDF model outperformed the rest of the systems in term of Accuracy.

Table 45 - Maximum and Minimum intervals of the systems against spam drift in term of Accuracy

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	70%	70%	40%	40%	40%	40%	30%	40%	40%	40%	40%

Min	60%	30%	40%	40%	40%	30%	10%	40%	30%	40%	20%

From the results above, we can conclude that ASDF is the most resilient model against spam drift problem, as it is consistently detecting spammers with high TP rate, Precision, Recall, F-Measure and Accuracy on the new dataset. The second best model is [6] as it can achieve the best TP rate, recall, F-measure and Accuracy based on the IBK classifier. Most of the subset of features used from the existing spam detection systems cannot handle the spam drift problem very well as most of the time it cannot detect any spammers. These results highlight the importance of utilising WEST for future researchers because this tool enable the researchers to quickly test and evaluate the performance of their system against the spam drift phenomenon.

4.4 Chapter summary

This chapter outlines the results of the experiments carried out for investigation studies on the features that have been proposed by the existing spammer detection systems. We used WEST to implement all the techniques surveyed, such as content-based and user-based features, number of recent tweets, feature selection algorithms and classifiers.

In this research, we have three research questions and have carried out several experiments. Based on our investigation results, we found a ASDF model (RQ1 and 2) that could effectively and efficiently identify spammers. Based on the results, it is outperforming the existing systems also: it took 3 minutes to build and classify. We also carried out another experiment (RQ3) with a new dataset and ASDF is detecting spammers at 91% TP rate. The existing systems however cannot handle spam drift well. Comparison of ASDF against the existing models showed that our model is outperforming the existing models.

In the next chapter, we will summarise the whole thesis, discuss the limitations of this research, find this research and also make improvements for the further research.

CHAPTER 5

CONCLUSION & FUTURE WORK

The objective of this research is to find the most effective features (RQ 1), efficient model (RQ 2) and a resilient system (RQ 3) to identify spammers on Twitter. The previous chapter showed the experimental results to find the most efficient, effective and resilient model based on the existing techniques and we found a model and called ASDF and this model achieved 91% TP rate. In the previous Chapter, we also compared the performance of the ASDF model against the existing systems in term of spam drift and according to the results, the ASDF model is outperforming to the rest of the systems.

This chapter summarises the research objectives, presents the significance of the final findings and discusses improvements that could be used for furthering the research.

5.1 Summary of the research objective

Twitter is a popular social networking platform released in 2006. Users interact by sending text-based messages, knows as *tweets.* A *tweet* is limited to 140 characters [5]. Its popularity attracts many spammers and about 83% of users have received at least one unwanted friend request or message on social networks [9]; about 3% of the tweets are spam, and 45% of users click on links posted by friends from their friend list's account, even though they do not know those people in real life [6]. The users identify spam manually based on their experience, thus this could lead to false positive problems and time wasting.

Many approaches have been proposed to distinguish between spammers and legitimate users such as [4, 6, 11, 22, 40]. It is also a challenge to determine the best system to identify spammers on Twitter, since all the existing systems employ different techniques, such as using different features, or extracting the features on different numbers of recent tweets and classifier(s). The best system is one which is able to efficiently identify spammers with a high degree of accuracy, where efficiency is a measure of the speed at which the system can detect spammers, and the time taken to extract the features. The factors that may affect these criteria are feature selection and classifiers(s). The feature selection criteria involve the time taken to extract the features, and how well the features perform in spammer detection. The classifier(s) criteria involve the time taken to build a model based on the selected features, selection of the best classifiers to identify spammers based on the proposed model, and the time taken to generate the results.

Furthermore, spammers keep evading current existing systems by changing the key features to disguise as legitimate accounts; this is known as spam drift. Also, most of the researchers did not evaluate their systems against spam drift, such as [15], [14].

In this research, we have investigated and identified a collection of current existing techniques that are effective, efficient and resilient at detecting spammers on Twitter based on three main

categories: best subset of features used to build the proposed model, the least number of recent tweets for feature extraction, and the best classifier to train the model. We also objectively compared the model that we found from answering the RQ 1 and 2 with the existing systems and evaluated the performance between them. It showed that this model outperformed the existing systems and coping well with spam drift with 91% TP rate (RQ 3).

In undertaking this research, we have defined the problem statement and the research questions as outlined in Chapter 1. The literature review regarding spam or spammer detection systems on Twitter, and also the current techniques to detect spam in other media platforms, such as YouTube or email, are examined in Chapter 2. Chapter 3 discussed the methodology used to build our proposed system, including how to extract the content-based and user-based features, and it details experiments for finding the least number of recent tweets, classifier(s) and evaluation. The experiments and evaluation results are analysed in Chapter 4 in order to answer the research questions.

5.2 Significance of Final Findings

The problem statement for the research has been answered, based on the results of the experiments carried out to find the most effective and efficient model to identify spammers in Twitter.

Problem Statement: "***What is the most effective, efficient and resilient model for detecting spammers on Twitter?"***

The best model for detecting spammers on Twitter is one that detects most accurately (measured in TP rate) in the shortest time possible. There are two parts were carried out to find the answer to the problem statement: the first part used to find the most effective features (RQ1) and most efficient model (RQ2) to build the ASDF. The second part used to find the resilient model (RQ3), it is testing whether the ASDF model found from RQ1 and RQ2 is able to cope with spam drift.

RQ1: ***"What are the most effective content-based and user-based features for detecting spammers on Twitter?"***

To find the most effective features, we used feature selection algorithms; InfoGain (we used ranking method to find top 1, 5, 10, 15, 20, 50 and 100 features) and CFSSubset to determine the most relevant subset of features for different numbers of recent tweets, such as 20RT, 50RT, 100RT, 150RT and 200RT. The results obtained showed the "CFS-100RT" subset of features extracted from 100RT is the optimisation subset of features to identify spammers, because it achieved a 100% TP rate, and Table 46 shows the optimisation subset of features from 100RT.

Table 46 - Best subset of features from 100RT

CFS-100RT subset of features from 100RT	
Number of mentions	Number of mentions
Number of exclamation marks	Number of exclamation marks
Number of spamwords	Number of spamwords
Number of spamwords in screenname	Number of spamwords in screenname

We also found the optimisation subset of features for each the number of recent tweets 20RT, 50RT, 150RT and 200RT. However, these subset of features did not perform well against 100RT in the experiments.

RQ2: ***"Which model is the most efficient at identifying spammers on Twitter?"***

The most efficient model to identify spammers is determined by the optimisation number of recent tweets and the best classifier. Because the "CFS-100RT" subset of features extracted from 100RT is the optimisation subset of features, we chose 100RT as the optimisation number of recent tweets for feature extraction, and RF classifier as the best classifier and also it is the fastest model compared to the existing systems. Therefore, "CFS-100RT" subset of features extracted from 100RT (RQ 1) contained the most effective features, and 100RT and RF classifier is becoming the efficient model (RQ 2) for identifying spammers on Twitter.

RQ3: ***"Which model is the most resilient at handling the spam drift phenomena?"***

We used the model that we have found in RQ1 and RQ2 to test its performance against spam drift problem. In this test, we used a latest dataset with tweets posted in 2015 and the results showed that ASDF model is able to detect at 91% TP rate, which means it is coping well with spam drift phenomena. Also, we tested the performance of the ASDF model against the existing systems on spam drift problem based on the common evaluation metrics. However, the results showed that their systems were not coping well with spam drift. While our ASDF model is achiving very well performance across all evaluation metrics. Thus, we believed ASDF is the most resilient model at handling the spam drift phenomena.

5.3 Summary and Future work

Twitter is a web application that allows users to post messages, called tweets, of up to 140 characters. Spammers exploit Twitter functions to spread malicious content. Therefore, identification of spammers is necessary to provide a clean environment for Twitter users. In this research, we studied the existing techniques to create a spammer detection system, such as features to identify spammers, a number of recent tweets for feature extraction and the classifiers to train the system. Based on these existing techniques, we found the most efficient, effective and resilient model to detect spammers on Twitter.

The main issues here are that many features have been proposed by different authors, and different numbers of recent tweets used to extract the features and different classifiers have been used to train the model. Therefore, it is difficult to evaluate the existing system's performance since they all use different techniques and spammers keep evading existing systems by changing their key features to disguise as a legitimate user; this phenomenon is called spam drift. In this research, we tried to find the most efficient, effective and resilient model to identify spammers. To do this, we collected 172 content-based and user-based features from different systems, used different numbers of recent tweets, such as 20RT, 50RT, 100RT, 150RT, and 200RT, and used two feature selection algorithms, such as InfoGain and CFSSubset, and five common classifiers (Naive Bayse, Support Vector Machine, Decision Tree, K-Nearest Neighbour, and Random Forest) to deploy many experiments in order to answer our research questions. Consequently, we needed a model

that could cope well with spam drift problem. Our ASDF model is tested and evaluated against the spam drift problem and this model could detect up to 91% TP rate in fact of spam drift.

We also performed the comparison between our model ASDF and the existing systems based on their performance against spam drift. The results showed that, the existing systems cannot handle spam drift well comparing to our model. The reason behind is that spammers were changing their behavior and the existing systems did not test their system against spam drift, while we did test the ASDF model against spam drift phenomena and this is why we believed our invented tool; WEST is helpful for future researchers to test their model. Furthermore, our model took only 3 minutes to build and classify, which is the fastest comparing to the existing systems.

Nevertheless, there is still room for improvement that can be continued in future work. We could improve the model by changing from content-based and user-based level to user-based and graph-based level, because graph-based focuses more on user behaviour and this is harder for spammers to disguise their account as a legitimate user, while content-based focuses on the linguistics of the tweets and it is requiring more time to process than graph-based and user-based. In fact, of identify spammers with faster time possible, we believed graph-based and user-based will be more efficient than content-based and user-based.

Also, we will improve the UI of WEST to optimise the experience for the users and provide more algorithms to WEST as the current version only contains some basic classifiers and algorithms for feature selection and classification processes.

To conclude, we investigated and studied a collection of current existing features to find an effective, efficient and a resilient model for spammers detection on Twitter and this model is coping well with spam drift and it achieved a 91% TP rate. Also, we believe that our research makes a significant contribution and provides substantial information to the community in the field, as well as expanding knowledge for future researchers.

REFERENCES

1 Gee, G., and Teh, H.: 'Twitter Spammer Profile Detection', 2010

2 Gilani, Z., Farahbakhsh, R., Tyson, G., Wang, L., and Crowcroft, J.: 'An in-depth characterisation of Bots and Humans on Twitter', arXiv preprint arXiv:1704.01508, 2017

3 Chen, C., Wang, Y., Zhang, J., Xiang, Y., Zhou, W., and Min, G.: 'Statistical features-based real-time detection of drifted Twitter spam', IEEE Transactions on Information Forensics and Security, 2017, 12, (4), pp. 914-925

4 Yang, C., Harkreader, R.C., and Gu, G.: 'Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers', in Editor (Ed.)^(Eds.): 'Book Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers' (Springer, 2011, edn.), pp. 318-337

5 Chu Zi, G.S., Wang Haining, Jajodia Sushil: 'Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?', IEEE Trans. Dependable Secur. Comput., 2012, 9, (November 2012), pp. 811--824

6 Mccord, M., and Chuah, M.: 'Spam detection on twitter using traditional classifiers', in Editor (Ed.) 'Book Spam detection on twitter using traditional classifiers' (Springer, 2011, edn.), pp. 175-186

7 Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V.: 'Detecting spammers on twitter', in Editor (Ed.)^(Eds.): 'Book Detecting spammers on twitter' (2010, edn.), pp. 12

8 Santos, I., Miñambres-Marcos, I., Laorden, C., Galán-García, P., Santamaría-Ibirika, A., and Bringas, P.G.: 'Twitter content-based spam filtering', in Editor (Ed.)^(Eds.): 'Book Twitter content-based spam filtering' (Springer, 2014, edn.), pp. 449-458

9 Stringhini, G., Kruegel, C., and Vigna, G.: 'Detecting spammers on social networks', in Editor (Ed.)^(Eds.): 'Book Detecting spammers on social networks' (ACM, 2010, edn.), pp. 1-9

10 Song, J., Lee, S., and Kim, J.: 'Spam Filtering in Twitter Using Sender-Receiver Relationship', in Sommer, R., Balzarotti, D., and Maier, G. (Eds.): 'Recent Advances in Intrusion Detection: 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011. Proceedings' (Springer Berlin Heidelberg, 2011), pp. 301-317

11 Chakraborty, A., Sundi, J., and Satapathy, S.: 'SPAM: A Framework for Social Profile Abuse Monitoring', in Editor (Ed.)^(Eds.): 'Book SPAM: A Framework for Social Profile Abuse Monitoring' (2012, edn.), pp.

12 Hirve, S., and Kamble, S.: 'Twitter Spam Detection', International Journal of Engineering Science, 2016, 2807

13 Feng, B., Li, Q., Pan, X., Zhang, J., and Guo, D.: 'GroupFound: An effective approach to detect suspicious accounts in online social networks', International Journal of Distributed Sensor Networks, 2017, 13, (7), pp. 1550147717722499

14 Alonso, O., Carson, C., Gerster, D., Ji, X., and Nabar, S.U.: 'Detecting uninteresting content in text streams', in Editor (Ed.)^(Eds.): 'Book Detecting uninteresting content in text streams' (2010, edn.), pp.
15 Bo Wang, Zubiaga, A., Liakata, M., and Procter, R.: 'Making the most of tweet-inherent features for social spam detection on Twitter', arXiv preprint arXiv:1503.07405, 2015
16 Sayce, D. (2017). Number of tweets per day? Retrieved from http://www.dsayce.com/social-media/tweets-day/
17 Wang, A.H.: 'Don't follow me: Spam detection in Twitter', in Editor (Ed.)^(Eds.): 'Book Don't follow me: Spam detection in Twitter' (2010, edn.), pp. 1-10
18 Ho, K., Liesaputra, V., Yongchareon, S., and Mohaghegh, M.: 'Evaluating social spammer detection systems', in Editor (Ed.)^(Eds.): 'Book Evaluating social spammer detection systems' (ACM, 2018, edn.), pp. 18
19 Ho, K., Liesaputra, V., Yongchareon, S., and Mohaghegh, M.: 'A Framework for Evaluating Anti Spammer Systems for Twitter', in Editor (Ed.)^(Eds.): 'Book A Framework for Evaluating Anti Spammer Systems for Twitter' (Springer, 2017, edn.), pp. 648-662
20 Lupher, A., Engle, C., and Xin, R.: 'Feature Selection and Classification of Spam on Social Networking Sites', in Editor (Ed.)^(Eds.): 'Book Feature Selection and Classification of Spam on Social Networking Sites' (2012, edn.), pp.
21 Amleshwaram, A.A., Reddy, N., Yadav, S., Gu, G., and Yang, C.: 'Cats: characterizing automation of twitter spammers', in Editor (Ed.)^(Eds.): 'Book Cats: characterizing automation of twitter spammers' (IEEE, 2013, edn.), pp. 1-10
22 Meda, C., Bisio, F., Gastaldo, P., and Zunino, R.: 'Machine Learning Techniques applied to Twitter Spammers Detection', 2014
23 Verma, M., and Sofat, S.: 'Techniques to Detect Spammers in Twitter-A Survey', International Journal of Computer Applications, 2014, 85, (10)
24 Chen, C., Zhang, J., Xiang, Y., and Zhou, W.: 'Asymmetric self-learning for tackling twitter spam drift', in Editor (Ed.)^(Eds.): 'Book Asymmetric self-learning for tackling twitter spam drift' (IEEE, 2015, edn.), pp. 208-213
25 Mittal, A.D.S.: 'Content Based Spam Classification in Twitter using MultiLayer Perceptron Learning', 2015, 5, (4)
26 Costa, J., Silva, C., Antunes, M., and Ribeiro, B.: 'Concept drift awareness in twitter streams', in Editor (Ed.)^(Eds.): 'Book Concept drift awareness in twitter streams' (IEEE, 2014, edn.), pp. 294-299
27 Indira, K., and Christal Joy, E.: 'Prevention of spammers and Promoters in Video Social Networks using SVM-KNN', International Journal of Engineering & Technology, 2014, 6, pp. 2024-2030
28 Singh, V.P.D.s.A.: 'A Novel Technique of Email Classification for Spam Detection', 2013, 5 - No. 10
29 Teli, S.P., and Biradar, S.: 'Effective email classification for spam and non-spam', 2014, 4, (6), pp. 273-308
30 Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Gonçalves, M.: 'Detecting spammers and content promoters in online video social networks', in Editor (Ed.)^(Eds.): 'Book

Detecting spammers and content promoters in online video social networks' (ACM, 2009, edn.), pp. 620-627
31 Nizamani, S., Memon, N., Wiil, U.K., and Karampelas, P.: 'Modeling Suspicious Email Detection using Enhanced Feature Selection', in Editor (Ed.)^(Eds.): 'Book Modeling Suspicious Email Detection using Enhanced Feature Selection' (2013, edn.), pp.
32 Sao, P., and Prashanthi, K.: 'E-mail Spam Classification Using Naïve Bayesian Classifier ', in Editor (Ed.)^(Eds.): 'Book E-mail Spam Classification Using Naïve Bayesian Classifier ' (2015, edn.), pp.
33 Shams, R., & Mercer, R.E. (2013, December). Classifying spam emails using text and readability features. In Data Mining (ICDM), 2013 IEEE 13th International Conference on (pp. 657-666). IEEE.
34 Jaswal, V., and Sood, N.: 'Spam detection system using hidden markov model', 2013, 3, (7)
35 Renuka, M.D.K.: 'Email classification for Spam Detection using Word Stemming', International Journal of Computer Applications, 2010, 1
36 Kondra Mohan Raju, E.M.: 'Classification of Users in Online Video Social Networks', International Journal of Innovative Technology and Exploring Engineering, 2013, 3, (7)
37 Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., Zhang, C., and Ross, K.: 'Identifying video spammers in online social networks', in Editor (Ed.)^(Eds.): 'Book Identifying video spammers in online social networks' (ACM, 2008, edn.), pp. 45-52
38 Burnap, P., Javed, A., Rana, O.F., and Awan, M.S.: 'Real-time classification of malicious URLs on Twitter using Machine Activity Data', in Editor (Ed.)^(Eds.): 'Book Real-time classification of malicious URLs on Twitter using Machine Activity Data' (ACM, 2015, edn.), pp. 970-977
39 Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., and Tesconi, M.: 'The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race', in Editor (Ed.)^(Eds.): 'Book The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race' (International World Wide Web Conferences Steering Committee, 2017, edn.), pp. 963-972
40 Lee, K., Caverlee, J., and Webb, S.: 'Uncovering social spammers: social honeypots+ machine learning', in Editor (Ed.)^(Eds.): 'Book Uncovering social spammers: social honeypots+ machine learning' (ACM, 2010, edn.), pp. 435-442
41 Lee, K., Eoff, B.D., and Caverlee, J.: 'Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter', in Editor (Ed.)^(Eds.): 'Book Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter' (2011, edn.), pp.
42 Galán-García, P., Puerta, J.G., Gómez, C.L., Santos, I., and Bringas, P.G.: 'Supervised Machine Learning for the Detection of Troll Profiles in Twitter Social Network: Application to a Real Case of Cyberbullying', in Herrero, Á., Baruque, B., Klett, F., Abraham, A., Snášel, V., Carvalho, C.P.L.F.A., Bringas, G.P., Zelinka, I., Quintián, H., and Corchado, E. (Eds.): 'International Joint Conference SOCO'13-CISIS'13-ICEUTE'13: Salamanca, Spain, September 11th-13th, 2013 Proceedings' (Springer International Publishing, 2014), pp. 419-428
43 Wang, A.H.: 'Detecting spam bots in online social networking sites: a machine learning approach': 'Data and Applications Security and Privacy XXIV' (Springer, 2010), pp. 335-342

44 Lin, P.-C., and Huang, P.-M.: 'A study of effective features for detecting long-surviving Twitter spam accounts', in Editor (Ed.)^(Eds.): 'Book A study of effective features for detecting long-surviving Twitter spam accounts' (IEEE, 2013, edn.), pp. 841-846
45 Opitz, D.W.: 'Feature selection for ensembles', in Editor (Ed.)^(Eds.): 'Book Feature selection for ensembles' (1999, edn.), pp. 384
46 Kira, K., and Rendell, L.A.: 'A practical approach to feature selection', in Editor (Ed.)^(Eds.): 'Book A practical approach to feature selection' (1992, edn.), pp. 249-256
47 Investopedia. (2018). Chi-square statistic. Retrieved from https://www.investopedia.com/terms/c/chi-square-statistic.asp
48 Hall, M.A.: 'Correlation-based Feature Selection for Machine Learning'. Doctor of Philosophy, The University of Waikato, 1999
49 Witten, I.H.: 'More Data Mining with Weka', in Editor (Ed.)^(Eds.): 'Book More Data Mining with Weka' (2014, edn.), pp.
50 T. Hamsapriya, D.K.R.a.M.R.C.: 'Spam classification based on supervised learning using machine learning techniques', 2011, 02, (04)
51 Brownlee, J.: '*8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset*', 2015
52 Chen, C., Zhang, J., Chen, X., Xiang, Y., and Zhou, W.: '6 million spam tweets: A large ground truth for timely Twitter spam detection', in Editor (Ed.)^(Eds.): 'Book 6 million spam tweets: A large ground truth for timely Twitter spam detection' (IEEE, 2015, edn.), pp. 7065-7070
53 To, S. H. (2018b). *T Test (Student's T-Test): Definition and Examples*. Retrieved from http://www.statisticshowto.com/probability-and-statistics/t-test/
54 To, S. H. (2018a). *ANOVA Test: Definition, Types, Examples*. Retrieved from http://www.statisticshowto.com/probability-and-statistics/hypothesis-testing/anova/
55 Minitab Inc: '*Why use an equivalence test?*', in Editor (Ed.)^(Eds.): 'Book *Why use an equivalence test?*' (2017, edn.), pp.
56 Brownlee, J.: 'Master Machine Learning Algorithms: Discover how They Work and Implement Them from Scratch', 2016
57 Friedl, M.A., and Brodley, C.E.: 'Decision tree classification of land cover from remotely sensed data', Remote sensing of environment, 1997, 61, (3), pp. 399-409
58 Bambrick, N. (2016). Support Vector Machines: A Simple Explanation. Retrieved from https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html
59 Brownlee, J. (2016). *Naive Bayes for Machine Learning*. Retrieved from https://machinelearningmastery.com/naive-bayes-for-machine-learning/
60 Donges, N. (2018). *The Random Forest Algorithm*. Retrieved from https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd
61 Polamuri, S. (2017). How the random forest algorithm works in machine learning. Retrieved from http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/
62 Twitter, I. (2016). *Twitter libraries*. Retrieved from https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries.html
63 Buntine, W., and Niblett, T.: 'A Further Comparison of Splitting Rules for Decision-Tree Induction', Machine Learning, 1992, 8, (1), pp. 75-85

64 Quiet Affiliate Marketing: 'Free First-name and Last-name Databases', in Editor (Ed.)^(Eds.): 'Book Free First-name and Last-name Databases' (2009, edn.), pp.
65 Largest Cities in the World. (2014). Retrieved from http://www.downloadexcelfiles.com/wo_en/download-excel-file-list-largest-cities-world
66 Forbes Global 2000 - 2014. (2014, 2014). Retrieved from http://download.macrofocus.com/treemap/Forbes%20Global%202000%20-%202014.xls
67 The Moz Top 500. (2016). Retrieved from https://moz.com/top500
68 Sharma, M.: 'English Dictionary in CSV format', 2016. Retrieved from http://www.bragitoff.com/2016/03/english-dictionary-in-CSV-format/
69 Shuford, S. A. (2014). 723 Bad words to blacklist. Retrieved from http://www.frontgatemedia.com/a-list-of-723-bad-words-to-blacklist-and-how-to-use-facebooks-moderation-tool/
70 Edit distance. (2016). Retrieved from https://en.wikipedia.org/wiki/Edit_distance
71 Perone, C.S.: 'Machine Learning: Cosine Similarity for Vector Space Models', in Editor (Ed.)^(Eds.): 'Book Machine Learning: Cosine Similarity for Vector Space Models' (2013, edn.), pp.
72 Vamsi, T. (2015). How to avoid looking like a spam user on Twitter? Retrieved from https://www.getspotlyte.com/blog/how-to-avoid-looking-like-a-spam-user-on-twitter.html

APPENDIX

20 Recent tweets

Table 47 shows the TP rate result for spammers between the different subset of features from the 20 recent tweets. The ANOVA results show that no signficant difference at 95% confidence level between the nine subset of features (P-value = 0.0175), therefore we did not perform the T-Test technique.

Table 47 - The results of the subset of features for 20 Recent Tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-20RT
NB	67%	0%	93%	93%	70%	80%	66%	67%	42%
SMO	38%	0%	0%	0%	45%	13%	26%	42%	10%
IBK	43%	26%	22%	69%	24%	48%	41%	42%	67%
J48	51%	22%	43%	67%	43%	75%	59%	71%	79%
RF	41%	30%	22%	71%	33%	61%	42%	42%	79%

Table 48 shows the equivalence testing results for "All Features" and "Top 1", and, according to the T-test results (P-value = 0.0025), there was a difference between the two subsets of the features. It can be seen that "All Features" performed better than "Top 1", because the maximum and minimum of intervals were higher than "Top 1".

Table 48 - Equivalence testing results for All Features and Top 1 of 20 recent tweets

	All Features	Top 1
Max	**59%**	30%
Min	**35%**	1%

Table 49 shows the equivalence testing results for "All Features" and "Top 5". According to the T-test results (P-value = 0.2479), there was no difference between the two subsets of features. Based on the maximum interval, "Top 5" had better performance than "All Features", but it was not very much higher than "All Features". However, the minimum of "Top 5" was very low compared to the minimum of "All Features", therefore, we considered "All Features" was better than "Top 5".

Table 49 - Equivalence testing results for all features and top 5 of 20 recent tweets

	All Features	Top 5
Max	**59%**	71%
Min	**35%**	0%

Table 50 shows the Equivalence testing results for "All Features" and "Top 10". According to the T-test results (P-value = 0.2435), there was no difference between two subsets of features. Based

on the maximum and minimum intervals, it can be seen that the maximum of "Top 10" was much higher than "All Features", although the minimum of "Top 10" was smaller than "All Features", but it was not a lot smaller than "All Features"; therefore, we consider "Top 10" performed better than "All Features".

Table 50 - Equivalence testing results for all features and top 10 of 20 recent tweets

	All Features	**Top 10**
Max	59%	**95%**
Min	35%	**25%**

Table 51 shows the Equivalence testing results for "Top 10" and "Top 15". According to the T-Test results (P-value = 0.1800), there was no difference between the two subsets of features. Based on the maximum and minimum intervals, we consider that "Top 10" had better performance than "Top 15" because the maximum interval of "Top 10" achieved 95% which was much higher than "Top 15" and the minimums of two subsets of features were about the same, therefore, "Top 10" was better than "Top 15".

Table 51 - Equivalence testing results for top 10 and top 15 of 20 recent tweets

	Top 10	Top 15
Max	**95%**	60%
Min	**25%**	25%

Table 52 shows the Equivalence testing results for "Top 10" and "Top 20". According to the T-Test results (P-value = 0.4109), there was no difference between the two subsets of features. Also, it can be seen that the maximum of "Top 10" was higher than "Top 20" and the minimum of "Top 10" was not much smaller than "Top 20". Thus, we consider "Top 10" performed better than "Top 20".

Table 52 - Equivalence testing results for top 10 and top 20 of 20 recent tweets

	Top 10	Top 20
Max	**95%**	81%
Min	**25%**	29%

Table 53 shows the Equivalence testing results for "Top 10" and "Top 50". According to the T-Test results (P-value = 0.2329), there was no difference between two subsets of features. Based on the Equivalence testing results, the minimum of "Top 10" was smaller than "Top 50, but there was not a very big difference between the two results. However, the maximum of "Top 10" was much higher than "Top 50", thus we considered "Top 10" better than "Top 50".

Table 53 - Equivalence testing results for top 10 and top 50 of 20 recent tweets

	Top 10	Top 50
Max	**95%**	62%

Min	**25%**	30%

Table 54 shows the Equivalence testing results for "Top 10" and "Top 100". According to the T-Test results (P-value = 0.3420), there was no difference between two subsets of features. Based on the Equivalence testing results, we considered "Top 10" better than "Top 100", because the maximum interval of "Top 10" was much higher than "Top 100", while the minimum of "Top 10" was not much smaller than "Top 100". Thus, we consider "Top 10" had better performance than "Top 100".

Table 54 - Equivalence testing results for top 10 and top 100 of 20 recent tweets

	Top 10	Top 100
Max	**95%**	67%
Min	**25%**	33%

Table 55 shows the Equivalence testing results for "Top 10" and "CFSSubset". According to the T-Test results (P-value = 0.4078), there was no difference between the two subsets of features. Based on the maximum interval, it can be seen that "Top 10" had a higher result than "CFSSubset", while the minimums of the two subsets of features were the same. Therefore, we consider "Top 10" had a better performance than "CFSSubset".

Table 55 - Equivalence testing results for top 10 and cfssubset of 20 recent tweets

	Top 10	CFSSubset
Max	**95%**	85%
Min	**25%**	25%

50 Recent tweets

Table 56 shows the TP rate results for spammers between the different subsets of features for 50 recent tweets. Based on the ANOVA results, there was no significant difference at 95% confidence level between the nine subsets of features (P-value = 0.0511), therefore we did not perform the T-Test.

Table 56 - The results of the subset of features for 50 Recent Tweets

	All Features	**Top 1**	**Top 5**	**Top 10**	**Top 15**	**Top 20**	**Top 50**	**Top 100**	**CFS-50RT**
NB	66%	0%	97%	92%	66%	66%	58%	75%	30%
SMO	45%	0%	0%	0%	50%	15%	37%	45%	22%
IBK	24%	22%	20%	62%	24%	42%	34%	38%	68%
J48	43%	26%	20%	61%	43%	67%	52%	48%	80%
RF	33%	22%	11%	65%	33%	57%	41%	34%	80%

Table 57 show the Equivalence testing results of the subset of features from 50 recent tweets. It can be seen that the maximum TP rate of "Top 10" was higher than the others. However, the minimum rate just higher than "Top 1" and "Top 5", but it was not very lower comparing to the other subset of features. Therefore, we consider the "Top 10" as the optimisation subset of features.

Table 57 - Equivalence testing results for the subset of features of 50 recent tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-50RT
Max	58%	26%	68%	**89%**	59%	70%	54%	64%	83%
Min	26%	1%	0%	**22%**	27%	28%	34%	31%	29%

100 Recent tweets

Table 58 shows the TP rate results for spammers between the different subsets of features for the 100 recent tweets. The ANOVA results show that there was a significant difference at 95% confidence level between the subset of features (P-value = 0.0005), therefore we need to perform the T-Test.

Table 58 - The results of the subset of features for 100 Recent Tweets.

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-100RT
NB	60%	0%	11%	81%	39%	53%	55%	66%	48%
SMO	68%	0%	0%	0%	14%	17%	49%	68%	31%
IBK	12%	15%	19%	100%	33%	100%	25%	17%	100%
J48	49%	24%	26%	100%	69%	100%	50%	52%	100%
RF	23%	15%	17%	100%	71%	100%	35%	30%	100%

Table 59 shows the Equivalence testing results for "All Features" and "Top 1". According to the T-test results (P-value = 0.0133), there was a signficant difference between the two subsets of features. It can be seen that "All Features" had a better performance than "Top 1", because the maximum and the minimum intervals of "All Features" were higher than "Top 1". Thus, we consider "All Features" better than "Top 1".

Table 59 - Equivalence testing results for all features and top 1 of 100 recent tweets

	All Features	Top 1
Max	**67%**	21%
Min	**18%**	0%

Table 60 shows the Equivalence testing results for "All Features" and "Top 5". According to the T-test results (P-value = 0.0210), there was a signficant difference between the two subsets of features. It can be seen that "All Features" had a better performance than "Top 5", because the

maximum and the minimum intervals of "All Features" were higher than "Top 1". Thus, we consider "All Features" as the optimisation subset of features.

Table 60 - Equivalence testing results for all features and top 5 of 100 recent tweets

	All Features	Top 5
Max	**67%**	24%
Min	**18%**	4%

Table 61 shows the Equivalence testing results for "All Features" and "Top 10". According to the T-test results (P-value = 0.0863), there was no signficant difference between the two subsets of features. It can be seen that "Top 10" had better performance than "All Features", because the maximum and the minimum intervals of "Top 10" were higher than "All Features". Thus, we consider "Top 10" as the optimisation subset of features.

Table 61 - Equivalence testing results for all features and top 10 of 100 recent tweets

	All Features	Top 10
Max	67%	**119%**
Min	18%	**33%**

Table 62 shows the Equivalence testing results for "Top 10" and "Top 20". According to the T-test results (P-value = 0.4669), there was no signficant difference between the two subsets of features. It can be seen that the maximum and the minimum intervals of "Top 10" and "Top 20" were about the same, but the maximum intervals of "Top 10" was slightly higher than "Top 20". Thus, we consider "Top 10" as the optimisation subset of features.

Table 62 - Equivalence testing results for top 10 and top 20 of 100 recent tweets

	Top 10	Top 20
Max	**119%**	111%
Min	**33%**	37%

Table 63 shows the Equivalence testing results for "Top 10" and "Top 50". According to the T-test results (P-value = 0.0682), there was no signficant difference between the two subsets of features. It can be seen that the maximum and the minimum intervals of "Top 10" was higher than "Top 50". Therefore, we consider "Top 10" as the optimisation subset of features.

Table 63 - Equivalence testing results for top 10 and top 50 of 100 recent tweets

	Top 10	Top 50
Max	**119%**	55%
Min	**33%**	30%

Table 64 shows the Equivalence testing results for "Top 10" and "Top 100". According to the T-test results (P-value = 0.1062), there was no signficant difference between the two subsets of

features. It can be seen that the maximum and the minimum intervals of "Top 10" was higher than "Top 100". Therefore, we consider "Top 10" as the optimisation subset of features.

Table 64 - Equivalence testing results for top 10 and top 100 of 100 recent tweets

	Top 10	Top 100
Max	**119%**	69%
Min	**33%**	24%

Table 65 shows the Equivalence testing results for "Top 10" and "CFS-100RT". According to the T-test results (P-value = 0.1062), there was no signficant difference between the two subsets of features. It can be seen that the maximum interval "Top 10" was higher than "CFS-100RT", while the minimum interval of "Top 10" was not very low than "CFS-100RT". Therefore, we consider "Top 10" as the optimisation subset of features.

Table 65 - Equivalence testing results for top 10 and cfssubset of 100 recent tweets

	Top 10	CFS-100RT
Max	**119%**	108%
Min	**33%**	42%

150 Recent tweets

Table 66 shows the TP rate results for spammers between the different subset of features for the 150 recent tweets. The ANOVA results show that there was no significant difference at 95% confidence level between the subset of features (P-value = 0.3443), so it is not needed to perform the T-Test.

Table 66 - The results of the subset of features for 150 Recent Tweets.

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-150RT
NB	63%	0%	19%	61%	42%	67%	62%	61%	42%
SMO	33%	0%	0%	0%	14%	14%	12%	25%	22%
IBK	15%	90%	14%	13%	66%	48%	15%	15%	73%
J48	34%	20%	44%	37%	56%	49%	57%	52%	52%
RF	34%	90%	15%	40%	62%	57%	46%	30%	74%

Table 67 show the Equivalence testing results of the subset of features from 50 recent tweets. It can be seen that the maximum and the minimum interval of "Cfssubset-150RT" was higher than the others. Therefore, we consider the "CFS-150RT" as the optimisation subset of features.

Table 67 - Equivalence testing results for the subset of features of 150 recent tweets

	All Features	Top 1	Top 5	Top 10	Top 15	Top 20	Top 50	Top 100	CFS-150RT

Max	53%	86%	34%	54%	69%	66%	61%	55%	**74%**
Min	19%	0%	2%	6%	26%	27%	14%	17%	**30%**

200 Recent tweets

Table 68 shows the TP rate results for spammers between the different subsets of features from the 200 recent tweets. The ANOVA results show that there was a significant difference at 95% confidence level between the four subsets of features (P-value = 1.2315E-05), so we needed to perform a T-Test between each subset.

Table 68 - The results of the subset of features for 200 Recent Tweets

	All Features	**Top 1**	**Top 5**	**Top 10**	**Top 15**	**Top 20**	**Top 50**	**Top 100**	**CFS-200RT**
NB	60%	0%	13%	23%	39%	72%	58%	55%	41%
SMO	33%	0%	0%	0%	13%	13%	37%	32%	23%
IBK	16%	21%	10%	21%	53%	29%	34%	20%	79%
J48	54%	21%	15%	26%	56%	56%	52%	52%	58%
RF	28%	21%	10%	22%	63%	57%	41%	35%	71%

Table 69 shows the T-Test showed a significant difference (P-value = 0.0139) between "All Features" and "Top 1". We consider that "All Features" had a better performance than "Top 1 because the maximum and minimum of "All Features" was higher than "Top 1".

Table 69 - Equivalence testing results for All Features and Top 1 of 200 recent tweets

	All Feature	**Top 1**
Max	**57%**	12%
Min	**20%**	10%

Table 70 shows the results of Equivalence testing for “All Feature” and “Top 5”. The T-Test showed significant differences (P-value = 0.0049) between "All Features" and "Top 5". The maximum and minimum of "All Features" was higher than "Top 5". Thus, we consider that "All Feature" was better than "Top 5".

Table 70 - Equivalence testing results for All Features and Top 5 of 200 recent tweets

	All Feature	**Top 5**
Max	**57%**	15%
Min	**20%**	30%

Table 71 shows the Equivalence Testing results for “All Feature” and “Top 10”. According to the T-Test results, there were differences (P-value = 0.0325) between "All Features" and "Top 10". The maximum and minimum of “All Feature” were much higher than “Top 10”, so we consider that "All Features" had better performance than "Top 10".

Table 71 - Equivalence testing results for All Features and Top 10 of 200 recent tweets

	All Feature	Top 10
Max	**57%**	28%
Min	**20%**	70%

Table 72 shows the Equivalence Testing results for “All Feature” and “Top 15”. According to the T-Test results (P-value = 0.3127), there was no difference between the two subsets. However, “Top 15” had a higher maximum and minimum interval than “All Feature”, therefore, we consider “Top 15” as the optimisation subset of features.

Table 72 - Equivalence testing results for All Feature and Top 15 of 200 recent tweets

	All Feature	Top 15
Max	57%	**63%**
Min	20%	**25%**

Based on Table 73, the T-Test results (P-value = 0.3524) showed that there was no difference between the two subsets of features. According to the Equivalence Testing results, “Top 15” achieved higher performance than “Top 20”. Thus, we chose “Top 15” as the optimisation subset of feature.

Table 73 - Equivalence testing results for Top 15 and top 20 of 200 recent tweets

	Top 15	Top 20
Max	**63%**	68%
Min	**25%**	22%

Based on Table 74, the T-Test shows that there was no difference (P-value = 0.4844) between "Top 15" and "Top 50". It can be seen that the maximum of “Top 15” was higher than “Top 50”, while their minimum was the same, thus we chose “Top 15” subset of feature for the next test.

Table 74 - Equivalence testing results for Top 15 and Top 50 of 200 recent tweets

	Top 15	Top 50
Max	**63%**	54%
Min	**25%**	34%

Table 75 shows the equivalence testing results (P-value = 0.3001) for “Top 15” and “Top 100” of 200 recent tweets. It can be seen that the maximum of “Top 15” was higher than “Top 100”, while they had the same minimum. Thus, we consider “Top 15” was better than “Top 100”.

Table 75 - Equivalence testing results for top 15 and top 100 of 200 recent tweets

	Top 15	Top 100
Max	**63%**	53%
Min	**25%**	24%

Table 76 shows the Equivalence testing results (P-value = 0.2481) for “Top 15” and “CfsSubset” of 200 recent tweets. The T-Test shows that there was no difference between them. But the “CfsSubset” had better performance than “Top 15” as the maximum and the minimum was higher than “Top 15”.

Table 76 - Equivalence testing results for top 15 and top cfssubset of 200 recent tweets

	Top 15	**CfsSubset**
Max	63%	**76%**
Min	25%	**32%**

Based on the T-Test results from Tables 54 to 61, the "CFSSubset" had better performance overall because the maximum and minimum interval was higher than the others. So we consider the "CFSSubset" to be the optimised subset of features for 200 recent tweets.

The comparison between ASDF model and existing systems

Precision

Table 77 shows the Precision of our system and the others. Based on the ANOVA results, it shows that there was a significant difference at 95% confidence level between the system (P-value = 0.0070), so it was necessary to perform T-Test.

Table 77 - Precision of our proposed system with the other systems

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
NB	17%	30%	0%	10%	37%	68%	38%	25%	28%	17%	30%
SMO	0%	72%	0%	0%	38%	80%	73%	0%	100%	0%	72%
IBK	100%	36%	56%	13%	53%	43%	50%	64%	44%	100%	36%
J48	100%	36%	20%	13%	56%	39%	49%	77%	56%	100%	36%
RF	100%	71%	67%	8%	65%	61%	76%	88%	69%	100%	71%

Table 78 shows maximum and the minimum interval Precision of the systems. Our system achieved maximum 113% precision, which was a higher result than the others. But the minimum was lower compared to most others. However, our maximum precision was generally much higher than the others, therefore we consider our system outperformed them in term of precision.

Table 78 - Equivalence Testing results of Precision

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
Max	**113%**	69%	75%	14%	61%	75%	73%	87%	86%	27%	75%

Min	**12%**	28%	3%	3%	37%	41%	40%	13%	32%	5%	31%

Recall

Table 79 shows the Recall of our system and the others. Based on the ANOVA results, it shows significant differences at 95% confidence levels between the systems (P-value = 0.0003), so it was necessary to perform T-Test.

Table 79 - Recall of our proposed system with the other systems

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
NB	81%	25%	0%	11%	62%	30%	54%	36%	35%	43%	34%
SMO	0%	14%	0%	0%	38%	10%	22%	0%	13%	0%	13%
IBK	100%	32%	54%	14%	27%	42%	36%	55%	41%	22%	34%
J48	100%	46%	46%	17%	65%	36%	61%	76%	57%	27%	46%
RF	100%	40%	59%	4%	34%	39%	35%	73%	45%	14%	66%

Table 80 shows the maximum and minimum interval Recall of the systems. It can be seen that our system had the highest maximum and minimum compared to the other systems. Therefore, in term of Recall, we consider that our system had a better performance than other existing systems.

Table 80 - Equivalence Testing results of Recall

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
Max	**119%**	43%	61%	16%	62%	44%	57%	79%	54%	37%	57%
Min	**32%**	18%	2%	2%	28%	18%	25%	16%	21%	50%	19%

F-measure

Table 81 shows the F-Measure of our system and the others. Based on the ANOVA results, it shows significant differences at 95% confidence levels between the systems (P-value = 0.0001), so we have used T-Test and the results show that only Alonso's system was significantly different to our system (P-value = 0.03).

Table 81 - F-Measure of our proposed system with the other systems

	ASDF	**[6]**	**[1]**	**[14]**	**[7]**	**[43]**	**[15]**	**[40]**	**[44]**	**[11]**	**[10]**
NB	28%	28%	0%	10%	46%	41%	45%	30%	31%	16%	31%
SMO	0%	24%	0%	0%	38%	18%	33%	0%	22%	0%	19%
IBK	100%	34%	55%	14%	35%	43%	42%	59%	43%	22%	46%
J48	100%	41%	57%	15%	60%	37%	55%	77%	56%	27%	49%
RF	100%	51%	63%	6%	45%	47%	48%	80%	55%	17%	73%

Table 82 shows the maximum and minimum interval F-Measure of the systems. The maximum F-Measure of our system achieved 113%, which was much higher than the others. But our minimum achieved only 17%, which was lower than most other systems. However, our maximum interval is 113%, while the other systems were about 15% to 83%, which was much lower than our proposed system, therefore we consider our proposed system outperformed the others.

Table 82 - Equivalence Testing results of F-Measure

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	**113%**	46%	67%	15%	54%	48%	52%	83%	56%	26%	63%
Min	**17%**	24%	2%	2%	35%	25%	36%	15%	26%	6%	23%

Accuracy

Table 83 shows the Accuracy of our system and the others. Based on the ANOVA results, it shows no significant difference at 95% confidence level between the systems (P-value = 0.5326), so there was no T-Test required. The SMO achieved 88% accuracy. However, it was still lower than the other systems. The IBK, J48 and RF still performed very well compared to the other systems.

Table 83 - Accuracy of our proposed system with the other systems

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
NB	51%	90%	89%	84%	88%	91%	89%	88%	89%	65%	83%
SMO	88%	93%	89%	92%	90%	91%	93%	93%	93%	91%	88%
IBK	100%	90%	90%	85%	92%	89%	92%	94%	92%	87%	91%
J48	99%	90%	92%	84%	93%	88%	91%	96%	93%	88%	90%
RF	100%	94%	92%	88%	93%	91%	93%	97%	94%	89%	94%

Table 84 shows the equivalence testing results for Accuracy of the systems. The ASDF achieved maximum of interval is 100% while the minimum was only 66%. However, it seems like [40] approach had better performance than our approach because the maximum accuracy for [40] was 97%, which was almost 100% and the minimum was 90%, which was much higher than the ASDF. Therefore, in term of accuracy, [40] approach performed better than the ASDF.

Table 84 - Equivalence Testing results for Accuracy

	ASDF	[6]	[1]	[14]	[7]	[43]	[15]	[40]	[44]	[11]	[10]
Max	100%	93%	91%	90%	93%	91%	93%	**97%**	94%	94%	93%
Min	66%	89%	88%	83%	89%	88%	89%	**90%**	90%	73%	85%

Unitec
Institute of Technology
TE WHARE WANANGA O WAIRAKA

Full name of author: .....Trung Minh Ho (Kenny).....

ORCID number (Optional): .....

Full title of thesis/dissertation/research project *('the work')*:

Evaluating Spammer detection systems for Twitter

Practice Pathway: Computer Science

Degree: Master of computing

Year of presentation:2018

Principal Supervisor: Dr. Veronica Liesaputra

Associate Supervisor: Dr. Sira and Dr. Mahsa.....

Permission to make open access
I agree to a digital copy of my final thesis/work being uploaded to the Unitec institutional repository and being made viewable worldwide.

Copyright Rights:
Unless otherwise stated this work is protected by copyright with all rights reserved.
I provide this copy in the expectation that due acknowledgement of its use is made.

AND

Copyright Compliance:
I confirm that I either used no substantial portions of third party copyright material, including charts, diagrams, graphs, photographs or maps in my thesis/work or I have obtained permission for such material to be made accessible worldwide via the Internet.

Signature of author: Trung

Date: ...06.... /.....08.../...2018.....

Unitec
Institute of Technology
TE WHARE WĀNANGA O WAIRAKA

Declaration

Name of candidate: Trung Minh Ho.....

This Thesis/Dissertation/Research Project **entitled** : Evaluating spammer detection systems for Twitter
is submitted in partial fulfillment for the requirements for the Unitec degree of
...Master of Computing..
Principal Supervisor: _Dr. Veronica Liesaputra_

Associate Supervisor/s: _____

CANDIDATE'S DECLARATION

I confirm that:

- This Thesis/Dissertation/Research Project represents my own work;
- The contribution of supervisors and others to this work was consistent with the Unitec Regulations and Policies.
- Research for this work has been conducted in accordance with the Unitec Research Ethics Committee Policy and Procedures, and has fulfilled any requirements set for this project by the Unitec Research Ethics Committee.
  Research Ethics Committee Approval Number: .....

Candidate Signature: Trung Date: 04/02/19

Student number: 1396701