Near Real-time Twitter Spam Detection with Machine Learning Techniques

Nan Sun\textsuperscript{a}, Guanjun Lin\textsuperscript{a}, Junyang Qiu\textsuperscript{a}, Paul Rimba\textsuperscript{b}

\textsuperscript{a}School of Information Technology, Deakin University, Geelong Victoria 3216, Australia; \textsuperscript{b}Data61, CSIRO, Melbourne Victoria 3008, Australia

ARTICLE HISTORY
Compiled September 2, 2017

ABSTRACT
The popularity of social media networks, such as Twitter, leads to an increasing number of spamming activities. Researchers employed various machine learning methods to detect Twitter spam. However, majorities of existing researches are limited to theoretically study, few of them can apply detection techniques to real-time scenario. In this paper, we bridge the gap by proposing a near real-time Twitter spam detection system, which provides near real-time tweets data acquisition, light-weight features extraction from a specific Twitter account, training detection model and online visualizing detection results. In this system, account-based and content-based features are extracted to facilitate spam detection. The models that are applied to our Twitter spam detection system are trained based on 1.5 million public tweets and nine mainstream algorithms. In addition, in order to efficiently reduce training time spent on massive data and save the cost of model updating, a parallel computing technique is introduced to train and update the models in this system. Empirical results verify that the model can achieve satisfactory performance based on our datasets. Furthermore, we implement a near real-time Twitter spam detection system which can better protect users from combating spams. This system also acts as a tweets collection tool, allowing researchers to test the performance of trained classifiers in realistic scenarios.

KEYWORDS
Social network security; Spam detection; Machine learning; Classification

1. Introduction

Nowadays, we have entered into the epoch of online social networks (OSNs). OSNs, such as Twitter, Facebook and Instagram, have occupied our life and have revolutionized the way that we socialize and communicate. Thanks to OSNs, we are able to communicate with our family, friends and colleagues anytime, anywhere. In addition, numerous companies have used OSNs as an advertisement and marketing tool to boost sales. Twitter is one of the most popular OSNs with tremendous popularity, it has up to 313 million active users according to the latest statistics [1].

On the one hand, the enormous growth of Twitter allows increasing number of users to share their information and get in touch with each other. On the other hand, the popularity of Twitter attracts spammers, leading to the spread of spams. Twitter spams usually refer to tweets which include advertisements, drugs sales or messages that redirect users to external malicious links. This may lead to phishing attack or
malware downloads, etc. [2]. Spams on Twitter not only affect the online social experience, but also threaten the security of cyberspace. For example, in September 2014, New Zealands’ network was melt down as the cause of a malware downloading spam [3], the result of which signalled the alarm of Twitter spams. Hence, the huge quantity and great threat Twitter spams are urgent to be prevented.

In order to effectively decrease spamming activities in Twitter, various Twitter spam detection approaches have been proposed, including one by Twitter itself [2], [4], [5], [6], [7]. To the best of our knowledge, there are three major categories of methods for Twitter spam detection. The first category method is based on user account and tweets content features [2] [5], [8]. For example, the number of followers/followings, the existence time of an account and the number of URLs contained in a tweet. These features are easily extracted from tweets with little or no computation. Nevertheless, these features can be easily fabricated. The second category relies on robust features derived from the social graph [4], [6], [7], which aims to explore the relationship of senders and receivers. However, graph-based features are empirically difficult to collect because generating a large social/relationship graph can be time and resources consuming. This is due to the fault that a user may interact with a large but unpredictable number of users. The third category focuses on tweets with URLs [8], [9], [10]. For example, domains and IP blacklists are widely used to filter tweets including malicious URLs. Nevertheless, this category is based on the assumption that malicious tweets embed URLs.

However, most of the existing researches emphasized on theoretical study. There is a lack of applicable Twitter spam detection systems that offer efficient online detection to combat spammers in real world. In order to bridge this gap, in this paper, we implement a near real-time Twitter spam detection system on the basis of empirical study on real world datasets.

To investigate the feasibility of achieving near real-time detection capability, we conduct a series of empirical studies by using 20k tweets, nine classification algorithms and various number of hosting CPU cores (1, 2, 4, 8, 16 and 32). Based on the results of the empirical studies, we propose a near real-time Twitter spam detection system prototype. In summary, the contributions of this paper are the following:

- we conduct an empirical study of nine widely-used machine learning algorithms to evaluate the performance in terms of detection performance and stability. In addition, we study the scalability of different algorithms to explore the efficiency of the algorithms running on parallel computing environment.
- We propose a near real-time Twitter spam detection system based on the empirical study. Under the circumstance of a small scale of parallel environment, the system is capable of achieving satisfactory performance, which can maintain above 80% detection accuracy.

The rest of this paper is organized as follows. The related work is presented in Section 2. Section 3 is a brief introduction of the process of data acquisition, and then, we address the light-weight features abstracted from data. In Section 4, we demonstrate the empirical study procedure and results. Based on these experiments’ results and analysis, we conceal the Twitter spam detection system from the system framework, key techniques and system implementation in Section 5. The conclusion and future work are in Section 6.
2. Related works

Twitter spam detection is an important topic in social network security. Many pioneer researchers have been dedicated to the exploration of spam detection on the OSNs. However, spammers employ various techniques to avoid being detected. Thus, researchers have to improve their methods to better detect the latest spammer behaviors. In this section, we discuss previous studies of Twitter spam detection methods as well as provide an overview of existing applications of Twitter spam detection.

2.1. Twitter spam detection methods

A series of approaches have been proposed based on different types of features. Some work relied on the user profile features and message content features to identify spams [2], [5], [8]; some proposed using graph-based features, typically the distance and connectivity of a social graph [4], [6], [7]; and some others relied on embedded URLs as the means of spam detection features [8], [9], [10].

2.1.1. Profile and content-based features

The user profile and message content based features can be simply extracted with little computation cost by using Twitter API. Therefore, it is practical and convenient to collect a large quantity of account information and sample messages for analyzing and researching.

In [8], the authors used the account and content based features to detect spam Tweets. For example, the account-based features include time period an account has existed, the number of followers and followings et al. In addition, the content-based features contain the number of hashtags and URLs embedded in the message et al. Apart from considering the account and content based features, [2] and [5] also took the user behavior features into account, such as user social behaviors. In [2], more detailed behavior-related features were considered, such as existence of spam words on the users nick name, posting frequency and number of tweets posted per day and per week.

2.1.2. Social graph-based features

Although the profile and content features can be collected conveniently, it is possible to fabricate and modify these features to escape detection [4]. With the purpose of solving fake features fabrication, some improvements attempt to make use of social graph based features for classifying spams.

Song et al. [4] proposed a novel spam filtering system, relying on the sender-receiver relationships. Based on the analysis of the distance and connectivity of the relation graph, the system predicts whether an account is a spammer or not. In addition, Yang et al. [6] designed some robust features, such as graph and neighbor-based features to cope with the spammers who are constantly evolving their techniques to evade the detection. They proved that their approach achieved higher detection rate and lower false positive rate compared to several previous works. Additionally, in [7], the authors combined graph-based and content-based features to facilitate the spam filtering.
2.1.3. **URLs-based features**

Even though detecting Twitter spam using social graph features can achieve satisfied performance, collecting these features is time and money consuming because the Twitter user graph is huge and complex. Instead, there are some work using embedded URLs in tweets to detect spams on Twitter under the assumption that all spam tweets contain URLs.

Thomas et al. [9] developed a real-time system for detecting spams by crawling URLs. They used features extracted from URLs, such as, the domain token, the path tokens and the URLs query parameters as detection criteria. Moreover, in [10], Lee et al. relied on the characteristics of the tweets URLs, such as the Correlated URL Redirect Chains, to develop a near real-time system for detecting suspicious URLs in tweets.

2.2. **Applications of Twitter spam detection**

In order to apply Twitter spam detection approaches to practical application instead of restricting to theory, researchers have made efforts to establish real-time Twitter spam detection system as well [9], [11], [12], [13], [14], [15]. An efficient real-time Twitter spam detection system not only can clean-up spam in Twitter, but also ensure a great user experience.

In [4], Song et al. proposed a novel spam filtering system based on the analysis of the distance and connectivity of the social relation graph. This system is able to differentiate whether an account is a spammer or not. Their method achieved a very high true positive rate which was up to 99.7%, but this method was not applicable in real-time detection for its unsatisfactory computational performance and some unrealistic assumptions. Moreover, [7] established a prototype to detect suspicious Twitter users. In this system, data was collected by proposed web crawler. In terms of features extraction, the authors made use of directed graph model to explore the relationship between unique friend and follower. Novel graph-based features and content-based features were extracted to achieve the spam detection. The system model was trained by Bayesian algorithms by using the extracted features. Consequently, it may not be practical to apply social graph based features to a real-time spam detection system.

Lee et al. [10] proposed a near real-time detection system used for Twitter suspicious URL. They discovered a characteristic which hackers usually reuse the limited information and resources. Hence, the URL redirect chains always use the same URLs. Based on this characteristic, they developed an approach to find redirect chains by using the shared URLs and decide whether the URL is suspicious or not. Through their evaluation, they show that the system can achieve high accuracy and performance. However, the shortened malicious URLs are easily falsified and this assumption that the hackers use the same URLs is doubtful.

Based on literature review, we find that there are a great number of spam detection methods, while, the number of Twitter spam detection system which can apply detection methods to realistic situation is limited. Hence, designing a Twitter spam detection system with good performance and high velocity is urgent and significant.
3. Data acquisition and processing

3.1. Raw data capture

This section explains the Twitter data acquisition procedure. To verify the effectiveness of trained classifiers in realistic scenarios, we pull the latest tweets from Twitter account in real-time. In terms of empirical study, we focus on gaining experience and statistical analysis from the experiments. Hence, huge size data is applied to well-designed experiments. Based on the experiments’ results from empirical study, we collect real-time Twitter data to detect, which ensures the practicality of the system.

3.1.1. Training data

In the empirical study, with the purpose of exploring the feasibility of setting up a real-time Twitter spam detection, we use a great number of data to design and conduct experiments. As the continuation and extension of the previous work [8], we use the same data set as the one described in [8].

Totally, we collect 600 million tweets, all of which contain URLs. Based on [16] and [17], the majority of spam tweets embedded URLs to attract victims with malicious purposes. Therefore, our research builds on the assumption that all spam tweets contain URLs with the purpose of luring users to external phishing sites or malware downloading.

With the help of Trend Micros Web Reputation Service (WRS) [18], we are able to identify 6.5 million spams tweets whose URLs are identified as malicious. In some works, researchers had to label malicious tweets manually based on Blacklisting service, such as Google SafeBrowsing. Thanks to Trend Micros WRS, Trend Micros WRS can automatically check whether the URLs embedded in tweets are malicious by using its large dataset of URL reputation records. We define tweets with malicious URLs as spams. Due to the Trend Micros WRS devoting to collecting latest URLs and maintaining the database of URLs, the URLs reputation list is up-to-date, which ensures the accuracy of labelling. We identified 6.5 million spam from 600 million tweets, and we have obtained up to 30 million labelled tweets data in total to form our ground truth dataset. Our experiment data was randomly selected from this labelled dataset.

3.1.2. Real-time pulling data

One of the motivations of setting up the Twitter spam detection system is to help users detect whether a tweet is a spam or not in real time. Hence, in the Twitter spam detection system prototype, the Twitter streaming API [19] is used to pull up Tweets from Twitter, which can collect 200 tweets each time from a specific user account. After pre-processing, these tweets can be used for detection.

3.2. Light-weight feature extraction

Before applying data to empirical study experiments and near real-time twitter spam detection system, the last thing to do is to pre-process data and extract features from pre-processed data. Thanks to LINQ to Twitter [20], the data could be easily pre-processing before extracting features. In this paper, we have extracted 13 features from our randomly selected ground truth data. As described in [8], little computation effort was required to build our ground truth data by extracting light-weight statistical
Table 1. Account-based and Content-based Features Description

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Account-based Features</strong></td>
<td></td>
</tr>
<tr>
<td>account_age</td>
<td>The age of an account</td>
</tr>
<tr>
<td>no_follower</td>
<td># of followers</td>
</tr>
<tr>
<td>no_following</td>
<td># of followings</td>
</tr>
<tr>
<td>no_userfavorites</td>
<td># of favourites the user received</td>
</tr>
<tr>
<td>no_lists</td>
<td># of lists in which the user is a member of</td>
</tr>
<tr>
<td>no_tweets</td>
<td># of tweets that has been posted by the user</td>
</tr>
<tr>
<td><strong>Content-based Features</strong></td>
<td></td>
</tr>
<tr>
<td>no_retweets</td>
<td># of times this tweet has been retweeted</td>
</tr>
<tr>
<td>no_tweetfavorites</td>
<td># of favourites this tweet received</td>
</tr>
<tr>
<td>no_hashtag</td>
<td># of hashtags in this tweet</td>
</tr>
<tr>
<td>no_usermention</td>
<td># of times this tweet being mentioned</td>
</tr>
<tr>
<td>no_urls</td>
<td># of URLs contained in this tweet</td>
</tr>
<tr>
<td>no_char</td>
<td># of characters in this tweet</td>
</tr>
<tr>
<td>no_digits</td>
<td># of digits in this tweet</td>
</tr>
</tbody>
</table>

features. Additionally, applying light-weight features to spam detection also reduces the computational complexity during the model training and testing processes, which makes real-time spam detection achievable. The extracted features can be categorized into two groups: user profile-based features and tweet content-based features as summarized in Table 1. We believe that if an account is controlled by a spammer, the tweets sent by this account would be spams. However, if an account initially belonged to a legitimate user and then was comprised by a spammer, the account could send out normal tweets and spams. Therefore, it is necessary to analyse both the user behaviours features and the tweet content features for deciding spams and non-spams.

A great number of account-based features and content-based features could be extracted from a Tweet. We need to ensure that the features extracted by us are the most discriminative ones, which can reduce the running time and achieve near real-time detection. Nevertheless, if the number of chosen features is too small, the performance of our system may not be satisfied. Therefore, in order to ensure the accuracy of detection and reduce training time, we choose six account-based features and seven content-based features finally.

The user profile-based features include: the time period of the account has existed, the number of followers, the number of followings, the number of favourites this user has received, the number of lists in which the user is a member of and the number of tweets this user has sent. These six features depict the behaviours of the user account. Take the feature no_following as an example, we suppose that spammers have more followings than common users in most cases. In addition, the tweet content-based features include: the number of times this tweet has been retweeted, the number of favourites this tweet received, the number of hashtags in this tweet, the number of times this tweet is mentioned, the number of URLs that is included in this tweet, the number of characters and the number of digits in this tweet. These features are closely related to the content of a tweet. For instance, a spammer may add more hashtags in a tweet, in order to attract more users to browse and click the malicious URL. In [21], the researchers investigated the feature characteristics by plotting Cumulative Distribution Function (CDF) of these features. Their plots demonstrate that the six account-based features and seven content-based features that we abstracted can be used as useful discerning features to detect Twitter spam. For example, they found that spammers are related to more lists compared with normal users. In addition, unsurprisingly, as
spammers are expected to attract more people to click their malicious links, they send more tweets than non-spammers. Generally, the 13 features have discriminative capability to detect Twitter spam [21].

4. **Empirical study on Twitter spam detection**

4.1. **Machine learning algorithms for spam detection**

Machine learning techniques, which provide computers the power of learning by extracting or filtering useful information or patterns from raw data, have been widely applied in diverse fields [22]. In our experiment, we utilize a large amount of labelled tweets (6.5 million) as ground truth data in order to train nine supervised machine learning algorithms for spam/non-spam tweets classification. The detection performance of nine mainstream algorithms will be compared for the purpose of identifying the best-performed algorithms on our light-weight features datasets. The selected algorithms can be classified into five groups as follow:

- Decision tree-based algorithms are one of the main categories of popular machine learning tools used for classification. From decision tree-based algorithms, we apply Random Forest (RF) [23] and C5.0 [24] algorithms in our experiments, considering these two algorithms achieved adequate performance in the past research.

- In terms of our dataset containing 13 features, we choose K nearest neighbors (kNN)-based algorithms for the reason of the appropriateness for data samples with small number of dimensions [25]. Besides kNN, weighted K nearest neighbors (k-kNN) [26] is selected as well, which is an improvement of kNN algorithm. K-kNN algorithm assigns corresponding weight to the votes of neighbours. Therefore, the neighbours in varied distance result adversely in the unclassified samples.

- Boosting algorithms are another main category of popular machine learning tools which have gained tremendous attentions in data mining and statistical fields. Stochastic gradient boosting machine (GBM) [27] and Boosted Logistic Regression (BLR) [28] are chosen to represent the boosting family among these algorithms.

- Among the Bayesian algorithms, the Naive Bayes (NB), being a classic probabilistic classifier which builds on the assumption that all features of data are probabilistically independent [29], is selected as a candidate algorithm for our experiments.

- The neural network (NN) and deep learning (DL) [30] (implemented using the deep neural networks architecture) are chosen in contrast to each other. At the time of writing, deep learning is the most popular machine learning algorithm which has achieved practical success in the fields of computer vision, speech recognition and natural language processing [31].

4.2. **Experiment results**

The motivation of carrying out the empirical study experiments is to explore the feasibility of setting up a near real-time Twitter spam detection system. By conducting the experimental study, the characteristics of each algorithm are better understood,
which contributes to raising the capability of setting up a near real-time Twitter spam detection system. In the empirical study, we conduct three groups of well-designed experiments to explore the performance, stability and scalability of nine algorithms. We randomly select samples from 30 million labelled dataset, different sizes of the training data ranging from 2k to 200k as shown in the dataset 1, 2 and 3 in Table 2. For both training and testing data, we kept the ratio of spams and non-spams to be 1:1. In terms of Testing data, we used randomly selected 100,000 spam and 100,000 non-spam (totally 200,000) tweets to test the performance of trained classifiers.

In conclusion, we summary the main results and findings of the experiments from algorithms’ performance, stability and scalability as follow.

4.2.1. Performance

In order to measure the performance of nine algorithms, we import accuracy, true positive rate (TPR), false positive rate (FPR) and F-measure as evaluation metrics. The equation 1 indicates the calculation of accuracy, which equals to the percentage of correctly identified cases (both spams and nonspams) in the total number of examined cases. TPR (also called the recall) is defined as equation 2, which indicates that the probability of authentic spams that are correctly identified as spams in the total number of authentic spams. The FPR refers to the proportion of non-spams incorrectly classified as spams in the total number of actual non-spams using the equation 3. The precision is defined as the ratio of the correctly classified spams using equation 4. Lastly, the F-measure is calculated by the equation 5.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)
\]

\[
TPR = \frac{TP}{TP + FN} \quad (2)
\]

\[
FPR = \frac{FP}{FP + TN} \quad (3)
\]

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (4)
\]

\[
F-\text{measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (5)
\]

The TPR, FPR and F-measure are recorded in Table 3, in addition, the change trend of accuracy is shown in Figure 1. By investigating these data, we find that both Random Forest and C5.0 outperform other algorithms in terms of detection accuracy, the TPR, the FPR and the F-measure. In addition, as the number of training data rise from 2000 to 200000, the accuracy of classification increases as well. Specifically,

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of Spam Tweets</th>
<th># of Non-spam Tweets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,000</td>
<td>1,000</td>
</tr>
<tr>
<td>2</td>
<td>10,000</td>
<td>10,000</td>
</tr>
<tr>
<td>3</td>
<td>100,000</td>
<td>100,000</td>
</tr>
</tbody>
</table>
Deep Learning algorithm also achieves approximately 80% of accuracy when trained with 200k training data, its TPR and F-measure values are above 80%.

4.2.2. Stability

In order to access how stable each algorithm performs in terms of detection accuracy, we apply the standard deviation (SD) to quantify stability. Each experiment is conducted for 10 times in order to test how the detection accuracy varies each time due to the random selection of the training samples. By recording the accuracy of each algorithm for 10 times on various sizes of data, we calculated the standard deviation value as equation 6 for every algorithm. A large SD value implies the fluctuation in detection accuracy and instability of the performance.

Table 3. TPR, FPR and F-measure on Dataset 1, 2, 3 (The highest TPR, the lowest FPR and the highest F-measure are bolded)

<table>
<thead>
<tr>
<th>Unit: %</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TPR</td>
<td>FPR</td>
<td>F-measure</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>kNN</td>
<td>62.29</td>
<td>38.26</td>
<td>62.11</td>
</tr>
<tr>
<td>GBM</td>
<td>79.11</td>
<td>20.74</td>
<td>79.17</td>
</tr>
<tr>
<td>C5.0</td>
<td><strong>80.76</strong></td>
<td><strong>80.87</strong></td>
<td><strong>80.60</strong></td>
</tr>
<tr>
<td>NN</td>
<td>77.35</td>
<td>24.74</td>
<td>76.54</td>
</tr>
<tr>
<td>BLR</td>
<td>73.22</td>
<td>28.80</td>
<td>72.43</td>
</tr>
<tr>
<td>RF</td>
<td>80.58</td>
<td>15.49</td>
<td><strong>82.19</strong></td>
</tr>
<tr>
<td>NB</td>
<td>80.22</td>
<td>50.41</td>
<td>69.40</td>
</tr>
<tr>
<td>k-kNN</td>
<td>79.86</td>
<td>25.14</td>
<td>77.92</td>
</tr>
<tr>
<td>DL</td>
<td><strong>75.77</strong></td>
<td><strong>73.80</strong></td>
<td><strong>75.93</strong></td>
</tr>
</tbody>
</table>

Figure 1. Detection Accuracy (%) of 9 Algorithms Using Dataset 1, 2 and 3
\[ s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}} \] (6)

Figure 2 shows the comparison of stability with dataset 1, 2 and 3. It is clear that Random Forest performs more stable than other algorithms do, when trained with 2,000 or 20,000 training data. The detection accuracies of Deep Learning and Naive Bayes fluctuate substantially due to varying its size of training data.

4.2.3. Scalability

The scalability is used to evaluate how the algorithms scale on the parallel environment. The parallel implementation enables algorithms to make full use of multiple CPUs to accelerate the calculation process by running tasks on these CPUs simultaneously, therefore saving the calculation time. To measure scalability, we use the term speedup to quantify how much performance gain can be achieved by a parallel algorithm compared with its corresponding sequential version [32]. The formula for calculating the speedup is defined as equation 7, where \( S(p) \) is the speedup value, and \( p \) denotes the number of processors used; \( T_1 \) refers to the execution time needed to run the parallel algorithm using a single processor; \( T_p \) is the execution time needed to run the parallel algorithm using \( p \) processors with the same problem size [33]. The analysis of speedup trends helps to understand the relationship between the performance gain (reflected as the decrease of training time) and the amount of computational resources involved.

\[ S(p) = \frac{T_1}{T_p} \] (7)
Figure 3 shows the scalability comparison for all the algorithms. From the comparison, we find that Deep Learning algorithm can effectively take advantage of parallel computational resources and achieves scalability. For most algorithms, when the number of CPU cores reached to 16, adding more CPU cores can only contribute to a negligible decline in training/detecting time. However, for Deep Learning, when increasing the number of CPU cores from 16 to 32, we observe a considerable decrease in its training time.

When using 20,000 training data to train Deep Learning algorithm on 32 CPU cores, it takes 5 seconds to finish training and 2 seconds for detecting with an 83% of detection accuracy. Deep Learning algorithm is possible for instant spam detection when using a large scale of parallelism.

In a scenario where the classifier training time is critical, logistic regression is a good choice because it takes only 3 seconds to finish training with 200,000 tweets data of using 32 CPU cores. The detection time for GBM algorithm is around 2 seconds for 200,000 testing tweets with 32 CPU cores, which makes it the preferred candidate for timely detection.
5. Near real-time Twitter spam detection system implementation

5.1. Key techniques

5.1.1. Machine learning

Machine learning is the core technique of Artificial Intelligence (AI) which provides computer capability of learning from raw data without being clearly programmed. Machine learning explores the algorithms which can learn from and make predictions, which is applied widely in data mining. Thanks to machine learning techniques, the algorithms are able to build models through learning the patterns of raw data. Then the learned patterns can be applied to new data for predictions without the need of human involvement. Unlike the traditional approaches, such as blacklist filtering method, our near real-time Twitter spam detection system uses the trained classification models for detection of the spam tweets.

5.1.2. Parallel computing

The account and content-based features are used for classifying spams from non-spams in this system. Although these features are conveniently extracted, there is a critical limitation existing to evade the detection, which is the account features and content-based features are conveniently fabricated. Besides, spammers can change these features frequently by updating their spamming techniques.

To deal with this demanding challenge, parallel computing technique is applied to this system. As spammers update their spamming techniques, we need to update
classification models in a short period. Parallel computing technique enables the spam detection models to adjust to new pattern of the spam tweets timely. Last but not least, parallel computing helps to reduce the prediction time which makes real-time detection achievable.

5.2. System framework

5.2.1. Tweets collection and feature extraction

In order to apply Twitter spam detection methods to practical application, the first thing to do is to grab the latest data from Twitter. For the sake of maximizing the component independence of the system and achieving real-time data acquisition, the tweets collection and extraction module are developed using the hierarchical design methodology.

The process of tweets collection and extraction is divided into 3 layers: presentation layer, business logic layer and data access layer. The UI interface and functions for data visualization are encapsulated in the presentation layer. The calling of Twitter API, the implementation of light-weight feature extraction is included in the business logic layer. The data access layer provides the interface for accessing the data exchange module.
which transforms database access logic to SQL queries. The layered design offers high cohesion and low coupling, improving the ease of maintenance and alteration in future.

5.2.2. System flow chart

With the guidance of the experiments, a Twitter spam detection system is implemented. The system offers real-time detection capability by using the pre-trained classifiers to further test and verify our experiment outcomes in real-world. In addition, the system also can be a tool for the latest tweets collection. As Figure 4, the flow chart shows that the system composes of C# based web application and Shiny framework.

5.2.2.1. C# based web application. The system provides a web-based portal for pulling tweets and extracting features. It allows users to pull the latest tweets from their time line with Twitter API. Then, users can detect the pulled tweets by using the classifiers which are trained using 1 million ground truth data. With a few clicks, the system will invoke the trained classifiers to identify the spams and vividly present the results using graphs as seen in the Figure 5.

5.2.2.2. Shiny framework. The calling of the above-mentioned algorithms to perform the training and testing is implemented using R programming language, and thanks to Shiny [34] which is a web application framework for R, we could enable users to query data, train the classifiers, and test the tweets data using interactive web applications.

The processed results are stored in the database which can be invoked by the web portal for visualization. The flow chart Figure 4 illustrates the work flow and the structure of the prototype system.
5.3. **System functions demonstration**

This section demonstrates the key functions of Twitter spam detection system. User can log in this web-based system by inputting user name, password and validation code. In addition, if a new user logs in, he/she can sign up an account. In case the user forgot his/her user name or password, the system provides password retrieved function. And then, when the user accesses the system, a homepage is displayed as Figure 5. On the left of the homepage is the function menu of Twitter spam detection system, including dashboard, data management and model management. The following section describes the dashboard, data management and model management functions in details.

5.3.1. **Dashboard**

This page provides users the overview of Twitter data, composing of Spam/Non-spam ratio, data pulling history and training time comparison. Firstly, a pie chart shows the Spam/Non-spam ratio which indicates the percentage of spam, non-spam and undetected tweets as shown in Figure 5. In order to present the data more specifically, if the mouse slips the pie chart, the specific number of spam, non-spam and undetected tweets are displayed. This function enables users to vividly view the previous detection results. The second part of Twitter data overview is data pulling history, which is demonstrated by histogram. Specifically, the horizontal ordinate is the date of data pulling action, and the vertical coordinate is the number of pulled data as shown in Figure 6. In the end, the dashboard gives the user a training time comparison by histogram. The horizontal ordinate is classifier category and the vertical coordinate is the training time. By observing training time comparison, users can choose the classifiers which would be used to detect Tweets as displayed in Figure 7.

5.3.2. **Data management**

5.3.2.1. **Data acquisition.** Data is acquired by using Twitter API, at most 200 Tweets can be grabbed from a specific user account to this system each time. At the same time, the content and account-based features are extracted, which will be used for detecting spam. After acquiring Tweets successfully, a tip is shown indicating the pulling of data from Twitter timeline, below the pull data function button as shown in Figure 6.
5.3.2.2. **Data display.** After the data is pulled and stored in the Twitter spam detection system, these tweets are ready to be detected. Before detection, the information of pulling history can be checked. The specific information is shown in Figure 8. Besides the basic Twitter information, the content and account based-features are displayed in the Twitter data display page.

5.3.3. **Model management**

5.3.3.1. **Spam detection.** During the process of spam detection, on the side of saving detection time, the models which have been trained in empirical study are used in this Twitter spam detection system. In Figure 10, it is clear that nine algorithms (kNN, k-kNN, Naive Bayes, Boosted Logistic Regression, Deep Learning, GBM, C5.0, Random Forest and Neural Network) as well as 3 orders of magnitude training data (1000, 10000 and 100000) can be chosen by users, according to users’ time and requirements. In case the condition that a user has no idea on which classifiers they would like to choose, the basic introduction of the algorithms is shown using the page as described in Figure 10 (c). In order to cope with the rapidly changing spamming activities, the models in this Twitter spam detection system are refreshed efficiently by utilizing parallel computing technique.

5.3.3.2. **Detection results.** After deciding which classifiers and trained model will be applied to detection, the pulled tweets are ready to be detected. The user need to wait for just a few seconds in accordance with chosen models, then the results would be shown on this page as shown in Figure 9, which achieves real-time detection. In this page, the system gives user an summary of the probability of this Tweet being spam combining with nine algorithms results, which ensures the accuracy of detection result.
Figure 10. Spam detection models in Twitter spam detection system.
6. Conclusion and future work

In conclusion, we set up a near real-time system based on the empirical study. In empirical study, we collect nine mainstream machine learning algorithms and study their performance in terms of performance, stability and scalability using different tweets datasets. In terms of performance, we apply these classification algorithms under different scenarios to evaluate their performance of detecting Twitter spams in terms of detection accuracy, the TPR, the FPR and the F-measure. We also investigate the performance stability of each algorithm to understand how random sampling and variation of testing data affect the detection performance. The outcome of our experiment shows that Random Forest and C5.0 stand out due to their superior detection accuracy, and Random Forest perform more stable than other algorithms. Moreover, to understand the cost-effectiveness of these algorithms in a parallel environment when dealing with a large amount of datasets, we examine the scalability of each algorithm using the different number of CPU cores, aiming to observe how parallel computational resources impact the algorithms’ training process and how much resource is suitable for a certain problem size. Based on our experiments, we observe a very satisfactory speedup trend achieved by Deep Learning, showing that it is capable of effectively utilizing parallel resources to accomplish near real-time training and testing tasks.

Based on these results on the experiments, we develop a near real-time Twitter spam detection system to test and verify our experiment outcomes in real-world. This system achieves the functions including pulling up latest Tweets and near real-time detecting Twitter spam. By utilizing parallel computing technique, the speed of detection is dramatically increasing. It provides the opportunity to make the system achieve near real-time. Moreover, the detection models in this system can be refreshed efficiently due to parallel computing, which makes the system has the capability to combat a large number of smart spammers.

There are some problems worthy of further study in the future. Firstly, it will be interesting to extend the current experiments to evaluate whether the performance of these algorithms can be further improved by using more tweets for training. Secondly, for the scalability tests, we would like to carry out experiments on laboratory-size computer clusters to explore whether the algorithms such as Deep Learning and Random Forest would achieve better scalability and performance in a large scale of parallel environment. Moreover, more work could be done by evaluating the performance of the algorithms such as Deep Learning using GPUs, for a GPU consists of much more processing cores than a CPU, which is capable of offering a larger scale of parallel computing architecture. Lastly, we are always willing to improve our near real-time Twitter spam detection prototype system as a handy tool for tweets collection and better spam detection.

Acknowledgement(s)

We thank Trend Micro for offering their Web Reputation Service for labelling the tweets to form the ground truth.
References