2016

Identifying Suspended Accounts In Twitter

Xiutian Cui
University of Windsor

Identifying Suspended Accounts in Twitter

By

Xiutian Cui

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2016

Identifying Suspended Accounts in Twitter

by

Xiutian Cui

APPROVED BY:

A. Hussein
Department of Mathematics and Statistics

L. Rueda
School of Computer Science

J. Lu, Advisor
School of Computer Science

May 13th, 2016

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Large amount of Twitter accounts are suspended. Over five year period, about 14% accounts are terminated for reasons not specified explicitly by the service provider. We collected about 120,000 suspended users, along with their tweets and social relations. This thesis studies these suspended users, and compares them with normal users in terms of their tweets.

We train classifiers to automatically predict whether a user will be suspended. Three different kinds of features are used. We experimented using Nave Bayes method, including Bernoulli (BNB) and multinomial (MNB) plus various feature selection mechanisms (mutual information, chi square and point-wise mutual information) and achieved F1=78%. To reduce the high dimensions, in our second approach we use word2vec and doc2vec to represent each user with a vector of a shot and fixed length and achieved F1 (73%) using SVM with RBF function kernel. Random forest works best with F1=74% on this approach.

AKNOWLEDGEMENTS

I would like to present my gratitude to my supervisor Dr. Jianguo Lu for his valuable assistance and support during the past three years. I cannot image I could graduate without his help.

I also would like to express my appreciation to Dr. Abdulkadir Hussein, Dr. Luis Rueda. Thank you all for your valuable comments and suggestions on this thesis.

Meanwhile, I would like to thank Zhang Yi for his great help on introducing word2vec and doc2vec methods to me.

Finally, I want to thanks to my parents, my wife and my friends who give me consistent help over the past three years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Twitter is an online social network that provides users to send and read 140-short messages named "tweets". It has already become one of the most-visited websites all over the world. According to alexa.com, Twitter ranks 9th in the world top websites. About 320 million users sending and reading tweets on Twitter every month and the total number of registered users has already been over 1 billion [1]. By using tweets, Twitter now has been considered as one of the fastest way to share information. Obviously, it also attracts spammers.

Spammers are defined as those who send unsolicited tweets (spam), especially advertising tweets, as well as repeatedly sending mass duplicate messages [2, 3]. Spammers are usually generated by computers and works automatically. Twitter also faces the same problem as the war between websites and spammers never ends. Twitter will suspend users once they detect the behaviors of users abnormally, such as sending spam or abusing tweets.

So it is important to analyze the suspended users to explore some methods to predict whether a user is spammer or not. Some approaches have been studied, including machine learning technologies [4, 5], URL blacklists [6, 7], and some spammer traps [8, 9].

However all these approaches faced some problems. Machine learning approaches have already been widely used to detect spam email. Compared to classify spammers, detecting spam email is easy because they can collect a huge number of spam email and then train a classifier on it. But for detecting spammers, thing changed a lot because we don't have a large dataset of spammers. In the work of [4], they collected only 355 spammers and the authors of [5] collected 77 spammers, which should be considered too small to draw the whole picture of spammers on Twitter.

Second, machine learning methods are based on the features of spammers. These features can be content attributes, which are extracted from the tweets sent by users;

and user account attributes, such as how many friend or follower he has. However, these features can be easily manipulated by spammers.

Other methods, such as blacklist or traps cannot work well all the time. Spammers can easily avoid blacklists and traps by changing the approach of sending spam message and the content because it is costless to generate new spammers.

In our work, we analyzed a large number of suspended users and proposed a spammer prediction method. Unlike the previous work, we collected tweets from 113,347 suspended users during 5 years.

Based on this dataset, we combined the traditional machine learning technologies and Paragraph Vector word embedding method to mapping tweets into vectors so that we can predict whether a user will be suspended or not. We tried to classify them by Naive Bayes classifier on n-gram models derived from the tweets. We also tried 4 different feature selection methods, Mutual Information, Pointwise Mutual Information, Weighted Averaged Pointwise Mutual Information and χ^2. These methods can rank the features by score so that we can know which features are important and which features are noise. By analyzing the classification results on different selected features of these feature selection methods, we found that almost half of unigram features are noise and 9/10 of bigram features are noise. After removing these features, we achieved 76.75 % accuracy and 78.54% F1 on using top 10^6 features selected by WAPMI.

We also tried different word embedding methods to convert users into vectors. We tried some classifiers on converted user vectors. When using SVM with RBF function kernel, we achieved 73.28% accuracy and 73.39% F1 on 1,000 dimension user vector trained by Paragraph Vector method. Although this result is lower compared to the result of classification on feature selected n-gram models, this result is useful because it only depends on a 1,000 dimension vectors.

After analyzing the characteristic of bad words using in suspended users, we found the number of bad words using users is as twice larger as that number of normal users. And we also introduced Badscore which can rank words by how close they are from bad words.

The remainder of this thesis is structured as follows: In chapter II, we review the previous works on spammer detection in OSNs. In chapter III, we address our spammer detection method in detail. In chapter IV, we applied our experiments on twitter suspended users dataset and tried 3 different model to encode words in tweets,

together with classifications and feature selections on it. Finally, in chapter VI, we summarize our work and give out the conclusions.

CHAPTER 2

Review of The Literature

In this chapter, we will review some previews studies about suspended users on twitter. By now, there are not many works analyzing the characteristic of suspended users. The only work we found is proposed by Thomas et al [10]. They collected and analyzed 1.1 million accounts suspended by Twitter. Their results show that 77% spammers are suspended in the first day of their tweets, which makes them hard to form relationships with normal users. Instead, 17% of them use hashtags and 53% of them use mentions to reach out to normal users.

Other works were focusing on analyzing spammers and trying to find a way to detecting spammers based on the extracted features [4, 11] or the relationships between spammers [11, 5].

In 2010, Benevenuto et al. [4] addressed a study on the spammers who focused on sending spam concluding the trending topics in Twitter. The main method they used is to collect user profiles and tweets, then classify them into two groups, spammer and non-spammer, by using Support Vector Machine (SVM). There are four steps in their approach, crawling user data, labeling users, analyzing the characteristics of tweet content and user behaviours and using a supervised classifier to identify spammers.

In the same year, Moh et al. [5] analyzed how much information gained from the friends and followers of one user. They also proposed a learning process to determine whether or not a user is spammer. There are two steps in this process. The first step is to train a categorization algorithm to distinguish between spammers and non-spammers on a set of basic user features. And the second step is to train a classifier to generate new features, which depend on a user's followers being spammers or non-spammers.

In 2012, Ghosh et al. [11] analyzed over 40,000 spammer accounts suspended by Twitter and found out that link farming is wide spread and that a majority of spammers' links are farmed from a small fraction of Twitter users, the social

capitalists, who are themselves seeking to amass social capital and links by following back anyone who follows them. And they proposed a ranking system, *Collusionrank*, to penalize users from connecting to spammers.

2.1 Approaches

The works which are using machine learning methods to detecting spammers are using nearly the same approaches. First they collected data from twitter, including tweets, user account attributes and user relationships. After collecting, they will extract features from these data and try to train classifiers on the extracted features to see whether the features can represent the users and how well the classifiers work.

In order to classify the users into spammers and non-spammers, they used supervised classifier. So they need to label one collection that contains spammers and non-spammers. In this paper they focused on the users who sent the tweets about trending topic, so they need to build one collection of users who sent topics of (1) the Michael Jackson's death, (2) Susan Boyle's emergence, and (3) the hashtag "#musicmonday". 8,207 users have been labeled manually, including 355 spammers and 7,852 non-spammers. They then randomly chose 710 non-spammers to reduce the number of non-spammers. Thus, the total size of labeled collection is 1,065 users.

To use machine learning algorithms, they then identified the attributes of users. The attributes are divided into two categories: content attributes and user behavior attributes. Content attributes are the ones represented in what the users posted. User behavior attributes are the properties of the users' acting on Twitter. Both of these two kinds of attributes are shown in Table 1.

Category	Attribute
Content Attributes	number of hashtags per number of words on each tweet
	number of URLs per words
	number of words of each tweet
	number of characters of each tweet
	number of URLs on each tweet
	number of hashtags on each tweet
	number of numeric characters (i.e. 1,2,3) that appear on the text, number of users mentioned on each tweet
	number of times the tweet has been retweeted (counted by the presence of "RT @username" on the text)
User Behavior Attributes	number of followers
	number of followees
	fraction of followers per followees
	age of the user account
	number of times the user was mentioned
	number of times the user was replied to
	number of times the user replied someone
	number of followees of the users followers
	number tweets receveid from followees
	existence of spam words on the users screen name
	the minimum, maximum, average, and median of the time between tweets
	number of tweets posted per day and per week

TABLE 1: User Attributes in Benevenuto's work

After extracted features, they used SVM to classify user collections with the attributes that they identified in the previous section. The implementation of SVM they

used in their experiments is provided by libSVM. Each user in the user collection is presented by a vector of values, which contains the attributes of this user. SVM will first trains a model from the labeled user dataset, and then applies this model to the classify the unknown users into two classes: spammers and non-spammers.

In the work [5], the authors used almost same idea of [4] but they introduced a two steps categorization framework which can classify users not only based on the content and user behavior attributes, and it also relies on the user's friendships. The first step of this framework is to train a model based on manually labelled user collections. And then one extended attribute set will be generated for each user based on the predictions provided by the first learner and the user's position in the social network. The learner will then be trained on this extended attribute set.

Category	Attribute
Friend and Follower Attributes	follower-friend ratio
Basic Attributes	number of posts marked as favorites
Friend and Follower Attributes	friends added per day
Friend and Follower Attributes	followers added per day
Basic Attributes	account is protected?
Basic Attributes	updates per day
Basic Attributes	has url?
Basic Attributes	number of digits in account name
Friend and Follower Attributes	reciprocity.

TABLE 2: User Attributes in Moh's work

They extracted an attribute set for each user, which are shown in Table 2. Unlike the previous works, the authors took the friend follower relationship into consideration. They added some attributes which can measure the social network of users. For example, the reciprocity is the rate of how likely a user follows his followers. In practice, spammers tends to follow all the users who follows them. And they also added some new basic attributes such as the number of digits in account name, which has been proved useful in classification by Krause et al.[12].

The second step is to compute trust metric based on the classification result of

first step using extracted attribute set. The authors modified the original formula.

$$\text{trust metric} = \sum_{followers} \frac{1}{\#\text{users followed}}$$

They applied the following modifiers to this formula:

- **legit** accumulate only the values coming from users who are predicted to be legitimate users
- **capped** accumulate only values coming from up to 200 users
- **squared** use $\frac{1}{\#\text{users followed} \times \#\text{users followed}}$ instead of $\frac{1}{\#\text{users followed}}$

They tried the combinations of different classifiers on different steps. Then they calculated the accuracy, precision, recall, F1, and finally draw a Receiver Operating Characteristic Curve (ROC curve) to evaluate the test results of each combination.

Unlike these two papers which are focusing on detecting spammers on Twitter, the work of [11] studied the link farm formed by spammers on Twitter. The dataset they used includes a complete snapshot of the Twitter network and the complete history of tweets posted by all users as of August 2009 [13]. To identify the spammers in this dataset, they collected the user accounts which are suspended by Twitter. Although the primary reason for suspension of accounts is spam-activity, the accounts which are inactive for more than 6 months can also be suspended. One URL blacklist which contains the most popular URLs in spam tweets has been constructed to confirm that the suspended users are truly spammers. The authors fetched all the bit.ly or tinyurl URLs that were posted by each of the 379,340 suspended accounts and found that 41,352 suspended accounts had posted at least one shortened URL blacklisted by either of these two shortening services. These suspended accounts were considered to be spammers.

The authors studied how spammers acquire links to study link farm in Twitter by analyzing the nodes following and followed by the 41,352 spammers. They defined the nodes followed by a spammer as *spam-targets* and the nodes that follow a spammer as *spam-followers*. Spam-targets who also follow the spammer are called *targeted followers*. After computing the numbers of spammer-targets, spammer-followers and targeted followers, they found out that the majority (82%) of spam-followers have also been targeted by spammers. And targeted followers are likely to reciprocate most links from spammers. Top 100,000 spammer followers (rank based on the number of

links they created to the spammers) exhibited a reciprocation of 0.8 on average and created 60% links to the spammers.

The authors also computed the Pagerank of each user in this dataset and found out that by acquired large farm links from spammer followers, some of the rank of spammers are very high, 7 spammers rank within the top 10,000 (0.018% of all users) 304 and 2,131 spammers rank within the top 100,000 (0.18% of all users) and 1 million (1.8% of all users) users according to Pagerank, respectively.

The authors then analyzed the users who willing to reciprocate links from arbitrary users and the reason why they need to farm links. They plotted how the probability of a user reciprocating to a link from spammers varies with the user's indegree and found out that the lay users, who have low indegree, rarely respond to spammers. On the other hand, users with high indegree value are more likely to follower a spammer.

And the authors also found out that the top link farmers (top 100,000 spam-followers) sometimes are active contributors instead of spammers. The motivating factor for such users might be the desire to acquire social capital and thereby, influence.

The authors proposed Collusionrank, a Pagerank-like approach, to combat link farming in Twitter. Collusionrank algorithm can also be combined with any ranking strategy used to identify reputed users, in order to filter out users who gain high ranks by means of link farming. To evaluating Collusionrank, the authors computed the Collusionrank scores of all users in the Twitter social network, considering as the set of identified spammers S, a randomly selected subset of 600 out of the 41,352 spammers.

The result of evaluation showed the effect of ranking spammers of Collusionrank is great. While more than 40% of the 41,352 spammers appear within the top 20% positions in Pagerank, 94% of them are demoted to the last 10% positions in Collusionrank. Even when only a small set of 600 known spammers is used, this approach selectively filtered out from the top positions of Pagerank, most of the unidentified spammers and social capitalists who follow a large number of spammers.

2.2 Experiments

We compare the results from works [4, 5], which are trying to detect spammers based on machine learning approaches.

In [4], they collected all user IDs ranging from 0 to 80 million since August 2009, which have been considered as all users on Twitter since there is no single user in the collected data had a link to one user whose ID is greater than 80 million. Finally they collected 54,981,152 used accounts that were connected to each other by 1,963,263,821 social links, together with 1,755,925,520 tweets. Among those users, there are 8% accounts were set private and were ignored. The detail description of this dataset can be found on their project homepage[14].

They then trained SVM based on the features listed in Table 1. Table 3 shows the confusion matrix of classification result. About 70% of spammers and 96% of non-spammers were correctly classified. The Micro-F1 (which is calculated by first computing global precision and recall values for all classes, and then calculating F1) is 87.6 %.

		Predicted	
		Spammer	Non-spammers
True	Spammer	70.1%	29.9%
	Non-spammer	3.6%	96.4%

TABLE 3: Basic classification result in Benevenuto's work

To reduce the misclassifying of non-spammers, the authors used two approaches. First is to adjust J parameter in SVM. In SVM, J parameter can be used to give priority to one class over the other. With the varying of J, the rate of correctness of classify can be increased to 81.3% ($J = 5$), with the misclassifying of legitimate users has been increased to 17.9%.

The second approach they used is to reduce the size of attributes set. By sorting the attributes by their importance, the authors can remove the non-important attribute and give more weight to the important ones. They used two feature selection methods, information gain and χ^2, which are available in Weka. The results of these two methods are similar and the top 10 attributes in result are same. Table 4 shows the top 10 result of feature selection. And the result of classification when just using top 10 attributes instead of all attributes shows that top 10 attributes are enough to classify the users.

Rank	Attribute
1	fraction of tweets with URLs
2	age of the user account
3	average number of URLs per tweet
4	fraction of followers per followees
5	fraction of tweets the user had replied
6	number of tweets the user replied
7	number of tweets the user receive a reply
8	number of followees
9	number of followers
10	average number of hashtags per tweet

TABLE 4: Top 10 attributes in Benevenuto's work

The authors of [5] collected the account names of spammers using the web page twitspam.org, where users can submit the names of suspected spammers. Another part of spammers were added by the authors during they collected data. They obtained non-spammers from the users they followed. In total they collected one dataset that contains 77 spammers and 155 non-spammers. And for each user in this dataset, they also collected the information on up to 200 of their followers.

For there are two steps in classification, the authors tried different combination of classifiers. Then they calculated the accuracy, precision, recall, F1, and finally draw a Receiver Operating Characteristic Curve (ROC curve) to evaluate the test results of each combination. Table 5 and Table 6 show the evaluation metrics for RIPPER algorithm and C4.5 algorithm.

Metric	basic	basic+peer	peer	basic+trust	all features
Precision	0.79	0.80	0.75	0.88	0.84
Recall	0.84	0.83	0.71	0.85	0.85
F1	0.81	0.81	0.73	0.87	0.84
Accuracy	0.87	0.87	0.82	0.91	0.90

TABLE 5: Evaluation metrics for RIPPER algorithms with the different extended feature sets in Moh's work

Metric	basic	basic+peer	peer	basic+trust	all features
Precision	0.80	0.81	0.72	0.85	0.86
Recall	0.85	0.79	0.67	0.85	0.86
F1	0.83	0.80	0.69	0.85	0.86
Accuracy	0.88	0.87	0.80	0.90	0.90

TABLE 6: Evaluation metrics for C4.5 algorithms with the different extended feature sets in Moh's work

The authors also tried to measure the information provided by each features. To do so, the authors calculated the information gain and the chi square values for each feature in extended feature set.

The authors claimed that using RIPPER in two steps achieved the best performance among the combinations of classifiers. And top 10 features ranked by information gain and chi square value is shown in Table. 7 and Table. 8.

Attribute	Information gain
spammers to legit followers	0.48
friend-follower ratio	0.35
friends per day	0.34
trust metric legit.	0.34
trust metric legit. capped	0.29
trust metric	0.29
friend-follower average for friends	0.27
average protected for followers	0.25
trust metric legit. square	0.24
average protected for friends	0.24

TABLE 7: Top 10 Information gain values for data set extended with RIPPER in Moh's work. Added attributes are bold.

Attribute	Chi square value
spammers to legit followers	128.68
friends per day	106.72
trust metric legit.	105.49
friend-follower ratio	101.23
trust metric legit. capped	94.8697
trust metric	88.78
friend-follower average for friends	81.54
average protected for followers	80.57
trust metric legit. square	79.93
trust metric legit. square capped	74.99

TABLE 8: Top 10 χ^2 values for data set extended with RIPPER in Moh's work. Added attributes are bold.

2.3 Conclusion

Previous works studied a lot of spammers and trained classifier to detect spammers on Twitter. They found some extracted features based on the content attributes, account attributes and relationships of users can identify whether users are spammers. The classifiers they trained based on these features achieved a great results.

They also tried analyzing the structure in suspended users and behaviors of suspended users. They found that the suspended users lack of ways of form the social relationship with normal users, so they can only rely on mentions or hashtags to contact with normal users.

However, these works still have their own problems. First is the way they collected the spammers' data. The authors of [4] used the data of users who sent the tweets about trending topics, the authors of [5] used the data from twitspam.org and the authors of [11] used the data of suspended users who sent tweets including shortened URL. All the approaches should be considered can only show one part of suspended users. According to the analysis of our dataset, there are many suspended users who didn't send any tweets about trending topics or including shortened URL. And twitspam.org cannot provide a full list of suspended users because the users on that website are submitted by other users. In our work, on the other hand, we used the data

of all users who were suspended after 5 years, which can provide more information and full characteristics of suspended users.

Second problem is when analyzing the behaviors of suspended users, they didn't use full text of tweets. Whether users should be suspended, firstly and mainly is depending on the tweets they sent. So It is really significant to analyze the tweets of suspended users. But in their works, they only used some content attributes, such as number of hashtags per words or number of URLs per words. Such attributes can be easily manipulated by spammers by simply increasing the percentage of normal tweets in all tweets they send. What's more, shortened URL can be hidden by just remove http protocal header so that the blacklist system cannot detect the tweets containing urls.

Our work tried to used full text of tweets and large dataset of suspended users to avoid these problems.

CHAPTER 3

Dataset

The dataset we used in our experiments is collected by T. Xu et al [15]. There are 3,117,750 users' profile, social relations and tweets in this dataset. They used 4 machines with whitelisted IPs to crawl data by Twitter API. The crawling was first started with the most popular 20 users reported in [16] and then used snowball crawling strategy to crawl other users. The crawling period was from Mar. 6 to Apr. 2, 2010.

5 years later, 113,347 users in these dataset were suspended by Twitter. We randomly sampled 10% (11,334) of suspended users and the same number of normal users who are not in the suspended user set in the original dataset and combined them as our dataset. For each user in our dataset, we used regular expression to extract the tweets from the original dataset, resulting in 4,518,074 tweets from suspended users and 2,588,032 tweets from non-suspended users. The statistics of tweets of suspended and normal users are summarized in Table 9.

	Suspended Users	Non-Suspended Users
# Users	11,334	11,334
# Tweets	4,518,074	2,588,032
# Tokens	30,366,927	18,778,909
Vocabulary (# Unique Token)	1,489,281	1,089,437
Average Tweets Length	6.72	7.25
Average URL Rate	12.14%	11.90%
Average Mention Rate	27.43 %	23.39%
Average Hashtag Rate	4.32%	3.89%

TABLE 9: Statistics of tweet dataset

We analyzed some properties of our dataset and compared the results to show the differences between suspended users and non-suspended users. We started with the

number of tweets of each user. Fig. 1 shows the distribution the count of tweets from users. They follow power laws for both suspended and normal users. Most users have one or two tweets. Among suspended users, there are close to one thousand users who send tweets only once, while there are more than two thousand users who send tweets only once. Because of the scarcity of the text, these users will be difficult to classify. There are also some users who sent tweets close to two thousands. The maximal tweet number is two thousand, because the data are crawled with two thousand as a limit. Such distributions differ from most text corpora – in corpora such as Reuters data sets, document lengths follow normal or log-normal distributions, where most documents have medium length. In our data, most documents have very few tweets. This will make classification more challenging.

We can also find that suspended users tend to send more tweets, as the slopes intersect around 10 tweets. There are more suspended users who send tweets more than 10 times. In average, suspended user send 398.63 tweets, while normal users send only 228.34 tweets in average.

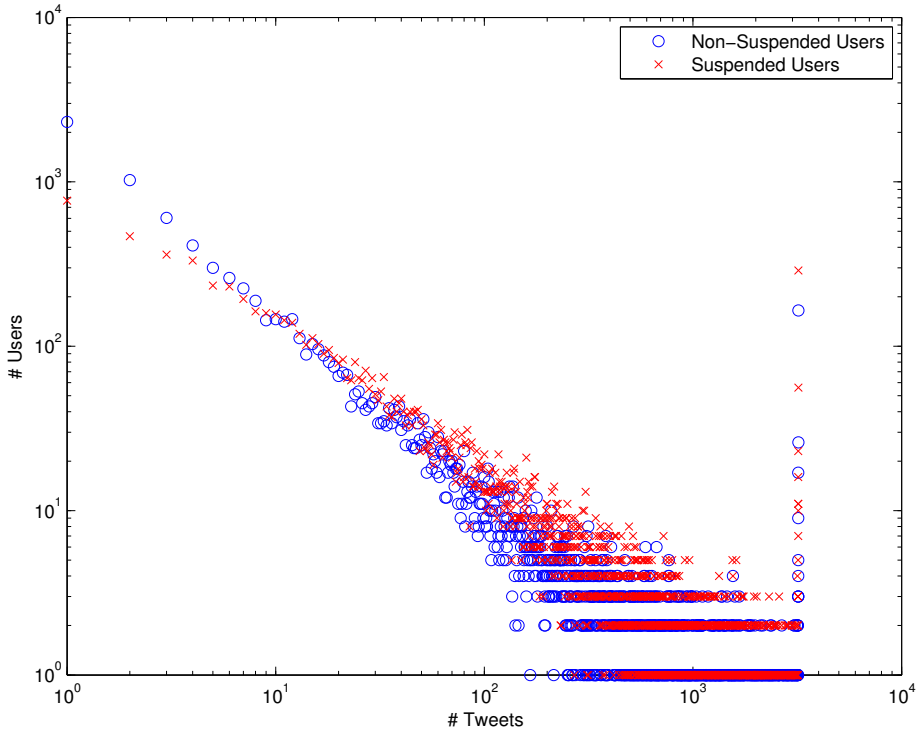

FIGURE 1: Tweet count distribution. Suspended users tend to have more tweets.

We then moved closer to look at the details of tweets by tokenizing the tweets into tokens. We distinguished several types of tokens in tweets, which are listed in

Table 10. We first used regular expression to split every tweet into tokens and then normalized the word tokens. Tokenization in tweets are different from normal text, because we need to keep track of urls, hashtags, and user mentioning.

Token Type	Description	Regular Expression	Example
Words	contains characters or digit numbers	[_A-Za-z0-9]+	Sample
Shortened URL	an url should start with http or https	http://[_A-Za-z0-9\./]+	http://t.co/ABcd123
Mention Users	used to mention other user by their username	@[_A-Za-z0-9]+	@twitter
Hashtags	used to mark keywords or topics in a tweet	#[_A-Za-z0-9]+	#spam

TABLE 10: Token Types in Tweets

After tokenizing, we turn all the tokens in lower case, then remove stop words using the stop word list from [17]. Stemming is also carried out using Porter2 stemming algorithms [18]. the program used in the experiment is downloaded from [19]. Stemming converts words into the root form so that different derived words from the same root will be treated as the same one. For example, after stemming "making" and "made", they will be converted into the root form "make".

After normalization, suspended user class contains total number of 30,366,927 tokens. Among them 1,489,281 are distinct. Normal users contain 18,778,909 tokens, 1,089,437 are distinct. These two vocabularies share 285,052 unique tokens in common.

The frequency distribution of tokens is plotted in Fig. 2. In both classes, the distributions follow Zipf's law as expected. There are very large number of terms that occur only once or twice. At the same time, there are also lot of popular terms that occur frequently. The slope is roughly two, consistent with most other text corpora. The counts for suspended class is higher because each user has more tweets.

After tokenizing, each tweet has been split into several tokens. We analysed the

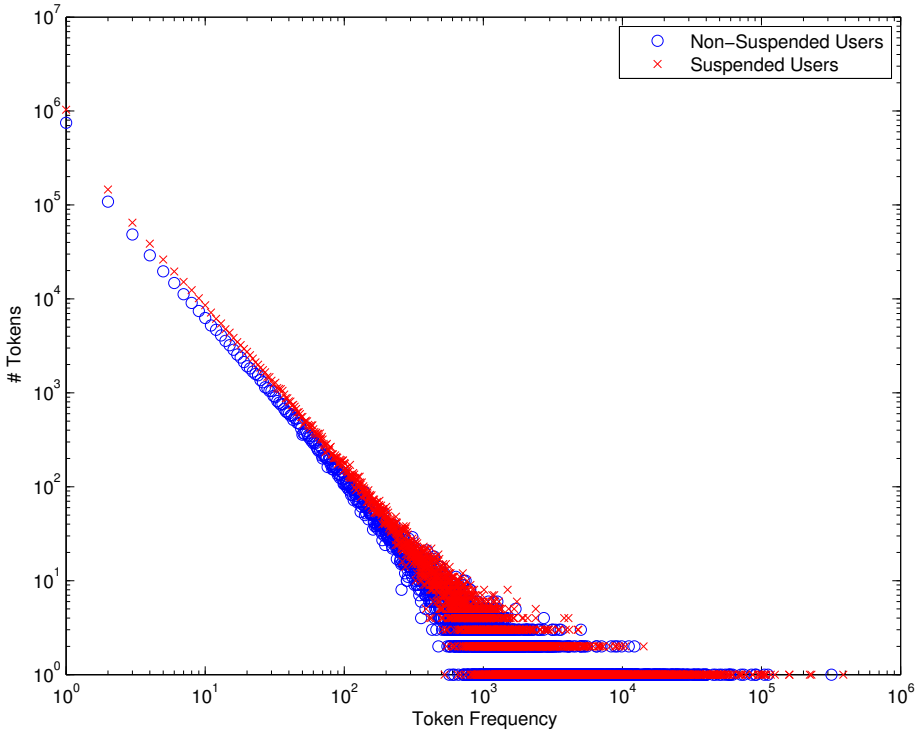

FIGURE 2: Frequency Distribution of Tokens

distribution of number of tokens in tweet. The average number of tokens in tweet from suspended users is 6.72 and that from non-suspended users is 7.25. Fig. 3 shows the distribution of number of tokens in tweet. This figure illustrates why the number of tweets from suspended users is larger than that of non-suspended users. This larger part is because of the number of short tweets (number of tokens < 10) from suspended users is much larger than that of normal users. This result and the result of tweets distribution can draw a conclusion that suspended users tend to send large number of short tweets. This conclusion can match our assumption that the main reason of suspending is because these users sending tweets against Twitter Rule, including abusive actives and spamming activities. Both of these two type of tweets are usually short on length while large on number to either abusing normal users or attracting normal users.

Among these tokens, URL, Mention and Hashtag are more special than the other tokens. The probabilities of occurrences of these 3 types of tokens have been plotted in Fig. 4. We excepted that the probabilities of these 3 types of tokens are very different between suspended users and non-suspended users as the previous studies [5, 4]. However, the results in Fig. 4 show that there is no big differences between

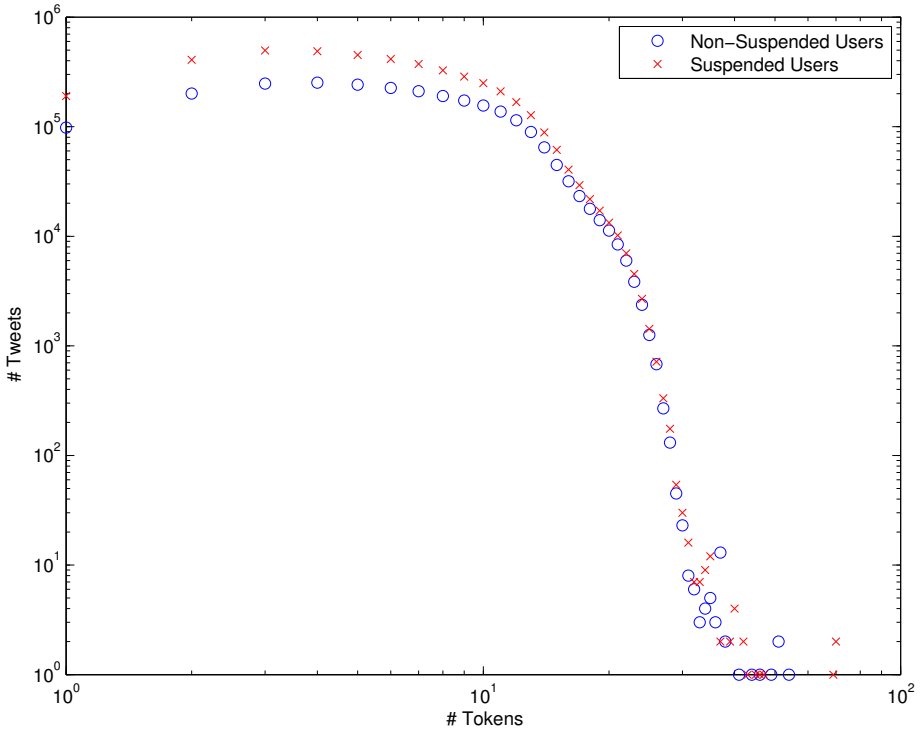

FIGURE 3: Number of Tokens Per Tweet Distribution

suspended and non-suspended users. The average of URL rate, mention rate and hashtag rate don't vary too much between these two dataset. The average rates are listed in Tabel 9.

What we can conclude from the analysis of suspended user dataset and non-suspended user dataset is that when using large, random sampled suspended user dataset instead of the target focusing crawling dataset in the previous studies, it is hard to classify users based on the rates of special tokens. The collecting methods that previous papers used cannot reveal the characteristics of suspending users because they were only focusing on a small group of users.

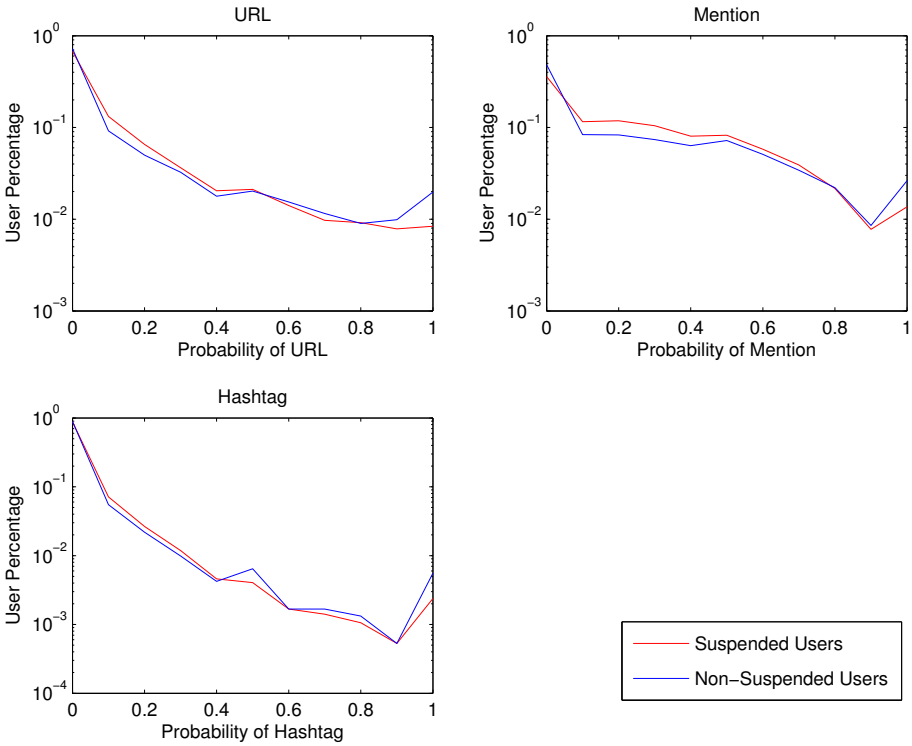

FIGURE 4: Distribution of Special Tokens

Summary we collected tweets of 113,347 suspended users, and tweets of equal number of normal users to avoid the complexity arising from imbalanced data. there are a few difference from other text corpora. In tokenization, we need to retain urls, mentioning, and hashtags. Document lengths follow power law instead of lognormal distributions. There are many very short documents. This will make classifications more challenging.

CHAPTER 4

Classification using N-gram Models

4.1 Naive Bayes Classifier

In this approach, all the terms are potential features, although we will also need to select from them for efficiency and performance consideration. This can be further divided into at least two models, the unigram model and bigram model.

In the unigram model, a document (i.e., all the tweets of a user) is treated as a bag-of-words. The word position information is discarded, while the count of word occurrences are retained. This model has the disadvantage that the order of the words are no longer relevant.

Thus, n-gram models are introduced. In n-gram models, a document is represented as a set of n-grams, where an n-gram is a consecutive sequence of terms with length n. Although in theory we can use tri-gram, or even 4-gram, in practice bigrams are most often used. Unlike unigram model, bigram model can carry a little information of word ordering. This is because when converting tweets into bigrams, the consecutive two words will be converted into one bigram.

In both unigram and bigram models, the feature size is very large, in the order of 10^6. Most classification methods can not run on such high dimension. Hence we experiment with Naive Bayes classifiers.

There are two different Naive Bayes classifiers, i.e., *Multinomial Naive Bayes Model* (MNB) and *Bernoulli Naive Bayes Model* (BNB). The difference between these two models is that Multinomial Model takes the number of occurrences into consideration while Bernoulli Model only considers weather a term occurs in user's tweets or not.

These two models are based on Bayes' theorem [20],

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where $P(A)$ and $P(B)$ are the probabilities of A and B; $P(A|B)$ is a conditional probability of observing event A given that B is true; $P(B|A)$ is the probability of observing event B given that A is true.

Hence, the probability of a user being suspended can be computed by,

$$P(c_s|w_1, w_2, ..., w_n) = \frac{P(c_s)P(w_1|w_2, ...w_n, c_s)P(w_2|w_3, ...w_n, c_s)...P(w_n|c_s)P(c_s)}{P(w_1, w_2, ..., w_n)}$$

where c_s is the event that user is suspended and w_k is the kth word in tweets from this user. After training, $P(w_1, w_2, ..., w_n)$ will be constant. So the probability $P(c_s|w_1, w_2, ..., w_n)$ is only depended on the prior probability $P(c_s)$ and the likelihood $P(c_s)P(w_1|w_2, ...w_n, c_s)P(w_2|w_3, ...w_n, c_s)...P(w_n|c_s)$. If using the assumption that the probability of each word are independent, which means the occurrences of words are not replying on others, $P(w_i|w_2, ...w_n, c_s) = P(w_i|c_s)$, we can simplify the formula based on this assumption,

$$P(c_s|w_1, w_2, ..., w_n) \propto P(c_s) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

So the classification result can be comparison of classes c which can maximize the probability $P(c|w_1, w_2, ..., w_n)$.

4.1.1 Multinomial Naive Bayes Model

In Multinomial Model, the class of a user can be determined by the following formula:

$$c = \arg\max_{c \in C} P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

where c is the classified class of this user, C contains two classes: Suspended User and Normal User, n_d is the number of tokens in user d, $P(c)$ is the probability of this user occurring in class c and $P(t_k|c)$ is the probability of token t_k occurring in a user of class c.

Users will be converted into vectors by their tweets under MNB. For example, there are two users whose tweets listed below:

1. User 1

(a) This is a sample tweet. see this: http://bitly.com/abc @user

(b) This is another tweet. #somehashtag

2. User 2

(a) This tweet contains some different words.

will be converted into two vectors by using unigram model:

$$S_1 = (1, 1, 1, 3, 1, 1, 1, 2, 2, 0, 0, 0, 0)$$
$$S_2 = (0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1)$$

The vocabulary and converting details are shown in Table. 11. All the words in this table are normalized so that they might be different from the original words in the tweets.

Unigram	# in User 1	# in User 2
#somehashtag	1	0
@user	1	0
anoth	1	0
see	1	0
this	3	1
sampl	1	0
http://bitly.com/abc	1	0
a	1	0
is	2	0
tweet	2	1
word	0	1
differ	0	1
some	0	1
contain	0	1

TABLE 11: Example of Unigram User Vectors

We estimated $P(c)$ and $P(t_k|c)$ by Maximum Likelihood Estimate (MLE).

$$P(c) = \frac{N_c}{N}$$
$$P(t_k|c) = \frac{T_{ct}}{T_c}$$

where N_c is the number of users in this class c, N is the total number of users, T_{ct} is the number of occurrences of token t in class c and T_c is the total number of tokens that occur in class c. In our cases, the number of suspended users and normal users are the same, so $P(c) = 0.5$. *Laplace smoothing* was used here to eliminate the condition that the number of token occurrence is 0, which is to add 1 to each term occurrences in the formula of $P(t_k|c)$:

$$P(t_k|c) = \frac{T_{ct} + 1}{T_c + |V|}$$

where $|V|$ is the size of vocabulary. Algorithm 1 illustrates how we train the Multinomial Model and use it to classify a user.

Procedure 1 Train Multinomial Naive Bayes Model and Classify Users

Input: labelled user feature map set U, class set C
Output: multinomial naive bayes model MNB_Model

procedure TRAIN_MULTINOMIAL_NB_MODEL
  $V \leftarrow$ EXTRACT_TOKENS(U)
  for $c \in C$ **do**
    $N_c \leftarrow$ COUNT_USER_IN_CLASS(U, c)
    $MNB\_Model.clsProb[c] \leftarrow log(\frac{N_c}{|U|})$
    for $t \in V$ **do**
      $T_ct \leftarrow$ COUNT_TOKENS_IN_CLASS(U, c, t)
      $T_c \leftarrow$ COUNT_TOKENS(U, c)
      $MNB\_Model.clsFeatureProb[c][t] \leftarrow log(\frac{T_{ct}+1}{T_c+|V|})$
    end for
  end for
  return MNB_Model
end procedure

Input: trained multinomial naive bayes model MNB_Model, unclassified user feature map u, class set C
Output: classified class c for u

procedure CLASSIFY_BY_MULTINOMIAL_NB_MODEL
  for $c \in C$ **do**
    $score[c] += MNB\_Model.clsProb[c]$
    for $k, v \in u$ **do**
      if $k \in MNB\_Model.clsFeatureProb[c]$ **then**
        $score[c] += MNB\_Model.clsFeatureProb[c][k] * v$
      end if
    end for
  end for
  return $\arg\max_{c \in C} score[c]$
end procedure

4.1.2 Bernoulli Naive Bayes Model

Bernoulli Naive Bayes Model, on the other hand, uses the boolean model in which the value of a token in user feature map in depending on whether this token occurs

in the tweets this user posted. The value of a token is 1 if the token occurs, otherwise it is 0. For example, the previous users will be converted into:

$$S_1 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0)$$
$$S_2 = (0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1)$$

The formula of classifying a user under Bernoulli Model is,

$$c = \arg\max_{c \in C} P(c) \prod_{1 \leq k \leq |V|} P(t_k|c)^{x_i}(1 - P(t_k|c)^{1-x_i})$$

where x_i is the boolean expression of whether token i occurs in the tweets of this user. The $P(c)$ and $P(t_k|c)$ can be estimated under Bernoulli Model like this,

$$P(c) = \frac{N_c}{N}$$
$$P(t_k|c) = \frac{N_{ct}}{N_c}$$

where N_{ct} is the number of users in class c whose tweets contain token t. We also smoothed the formula of $P(t_k|c)$:

$$P(t_k|c) = \frac{N_{ct} + 1}{N_c + 2}$$

Algorithm 2 illustrates how we train the Bernoulli Model and use it to classify a user.

Procedure 2 Train Bernoulli Naive Bayes Model and Classify Users

Input: labelled user feature map set U, class set C
Output: bernoulli naive bayes model BNB_Model

procedure TRAIN_BERNOULLI_NB_MODEL
  $BNB\_Model.V \leftarrow$ EXTRACT_TOKENS(U)
  for $c \in C$ **do**
    $N_c \leftarrow$ COUNT_USER_IN_CLASS(U, c)
    $MNB\_Model.clsProb[c] \leftarrow log(\frac{N_c}{|U|})$
    for $t \in BNB\_Model.V$ **do**
      $N_ct \leftarrow$ COUNT_USERS_CONTAINING_TOKEN_IN_CLASS(U, c, t)
      $BNB\_Model.clsFeatureProb[c][t] \leftarrow \frac{N_ct+1}{N_c+2}$
    end for
  end for
  return BNB_Model
end procedure

Input: trained bernoulli naive bayes model BNB_Model, unclassified user feature map u, class set C
Output: classified class c for u

procedure CLASSIFY_BY_BERNOULLI_NB_MODEL
  for $c \in C$ **do**
    $score[c] += BNB\_Model.clsProb[c]$
    for $t \in BNB\_Model.V$ **do**
      if $t \in u.keys$ **then**
        $score[c] += log(BNB\_Model.clsFeatureProb[c][t])$
      else
        $score[c] += log(1 - BNB\_Model.clsFeatureProb[c][t])$
      end if
    end for
  end for
  return $\arg\max_{c \in C} score[c]$
end procedure

4.2 Evaluation and Confusion Matrix

To evaluate the result of classification, we used N-fold cross validation. First divided the dataset into N folds, which are same size. And then run N times of validation on the datasets that one fold is used as test dataset and other folds are used as training dataset.

We used *confusion matrix* to visualize the result of cross validation. Table 12 shows the confusion matrix of 2 classes cross validation result, TP represents the number of suspended users that are classified as suspended users, FP represents the number of suspended users that are classified as normal users, FP represents the number of normal users that are classified as suspended users and FN represents the number of normal users that are classified as normal users. We can compute accuracy or F1 value using confusion matrix as well.

		Classified	
		Suspended User	Normal User
Actual Class	Suspended User	TP	FN
	Normal User	FP	TN

TABLE 12: 2 Classes Confusion Matrix Example

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$
$$Recall = \frac{TP}{TP + FN}$$
$$Precision = \frac{TP}{TP + FP}$$
$$F1 = 2 \times \frac{Precision + Recall}{Precision \times Recall}$$

These 4 parameters can measure the result of classification. Accuracy can measure the total accuracy rate of classification of suspended users and non-suspended users; recall shows the rate of how many users who are actually should be suspended will be suspended; precision can measure the precision of classifier on predicting suspending; F1 illustrates the overall performance of this classifier.

4.3 Feature Selection

When using n-gram language model, the main problems we face is the huge size of feature set, leading to a lot of time spending on training and testing. We can use feature selection algorithms to reduce the size of feature set. Another benefit we can get is that using feature selection can remove the irrelevant features, also known as noise features.

To remove the noise features and improve the result of classification, we used several feature selection algorithms: Mutual Information (MI), Pointwise Mutual Information (PMI), Weighted Average Pointwise Mutual Information (WAPMI) and Chi Square (χ^2). We computed the scores of each feature by these feature selection algorithms and then sort them by the scores.

Mutual Information, which is also called Information Gain, can measure how much information contribution of a token during the classification. We computed Mutual Information by the following formula,

$$MI(t,c) = \sum_{e_t \in 1,0} \sum_{e_c \in 1,0} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}$$

where e_t is a boolean variable representing whether term t occurs in user's tweets and e_c is a boolean variable that represents whether user is in class c. In our experiments, we let $e_c = 1$ represent the user is suspended and $e_c = 0$ represent user is normal user. We can also use MLE to estimate the probabilities $P(U = e_t, C = e_c), P(U = e_t), P(C = e_c)$:

$$\hat{P}(U = e_t, C = e_c) = \frac{N_{e_t e_c}}{N}$$
$$\hat{P}(U = e_t) = \frac{N_{e_t}}{N}$$
$$\hat{P}(C = e_c) = \frac{N_{e_c}}{N}$$

where N_{cond} is the number of users that match condition *cond*. For example N_{11} is the number of users that match two condition: t occurs in these users' tweets and all of these users are suspended. The formula of mutual information can be converted by using MLE estimation

$$MI(t,c) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}} + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}}$$

The variable meanings in this formula are listed in Table 13. We applied adding-one smooth to N_{11}, N_{01}, N_{10} and N_{00} to eliminate the problems that these numbers can be 0.

Variable	Meaning
N_{11}	the number of suspended users whose tweets contain term t
N_{01}	the number of suspended users whose tweets don't contain term t
N_{10}	the number of non-suspended users whose tweets contain term t
N_{00}	the number of non-suspended users whose tweets don't contain term t
$N_{1.}$	the number of users whose tweets contain term t
$N_{.1}$	the number of suspended users
$N_{0.}$	the number of users whose tweet don't contain term t
$N_{.0}$	the number of non-suspended users
N	the number of total users in both suspended user set and normal user set

TABLE 13: Meaning of N_{cond}

However, when the token frequency is imbalanced between suspended users and non-suspended users, although the number of suspended users ($N_{.1}$) and the number of non-suspended users are similar, we will still be facing the problem that the feature selection method will have more probability to select the features from suspended users than select from non-suspended users. We will show the result in experiment section about this. In order to solve this problem, we used token frequency instead of number of users. The definitions of N_{11} and N_{10}, which now are the frequency of this feature occurring in suspended user dataset and the frequency of this feature occurring in non-suspended user dataset, remain similar to the original definitions. And we can compute N_{01} and N_{00} by,

$$N_{01} = N_{.1} - N_{11}$$
$$N_{00} = N_{.0} - N_{10}$$

where $N_{.1}$ is the sum up of total feature frequency in suspended users and $N_{.0}$ is the sump up of total feature frequency in non-suspended users.

Expected value of feature frequency of t in class c can be computed by,

$$
\begin{aligned}
E_{11} &= N \times P(t) \times P(c) &&= \frac{N_{1.}N_{.1}}{N} \\
E_{10} &= N \times P(t) \times (1 - P(c)) &&= \frac{N_{1.}N_{.0}}{N} \\
E_{01} &= N \times (1 - P(t)) \times P(c) &&= \frac{N_{0.}N_{.1}}{N} \\
E_{00} &= N \times (1 - P(t)) \times (1 - P(c)) &&= \frac{N_{0.}N_{.0}}{N}
\end{aligned}
$$

So the formula of MI can be simplified to,

$$MI(t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{N_{e_t e_c}}{N} \log_2 \frac{N_{e_t e_c}}{E_{e_t e_c}}$$

We can easily figure out when 1) the frequency of this feature is high in both suspended users and non-suspended users; 2) the frequency of this feature is different between suspended users and non-suspended users slightly, MI will give this feature a high score. The first condition can prove to increase $P(U = e_t, C = e_c) = N_{e_t c_t}/N$ and the second condition can make $\frac{N_{e_t e_c}}{E_{e_t e_c}}$ increase. So together this feature will be selected by MI.

Pointwise Mutual Information is a little different from MI because MI is focusing on the average of all the events while PMI only is focusing on the single events:

$$
\begin{aligned}
PMI(t,c) &= \sum_{e_c \in \{0,1\}} |\log_2 \frac{P(U = e_t C = e_c)}{P(U = e_t)P(C = e_c)}| \\
&= \sum_{e_c \in \{0,1\}} |\log_2 \frac{N_{1e_c}}{E_{1e_c}}|
\end{aligned}
$$

Compared to the formula of MI, PMI only depends on the frequency difference between suspended users and non-suspended users, which is measure by $\frac{n}{e}$. So unlike that MI will select those popular features in both datasets, PMI will focus on the rare words instead.

Although MI and PMI can measure how strong the relationship between feature and the class is, there are 2 problems in them. First is that they all treat the features as independent random variables when they estimate the probability of features occurring in tweets. However, in real tweets the features (words) are not independent. This kind of estimation loses the relationship between features. Second is that when classifying using probability, such as using Multinomial Naive Bayes to classify

the tweets, the conditional probability usually is computed by merging all of tweets together as a large tweet. This will give the words in longer tweet a larger weight.

In 2005, K Schneider et al [21] proposed a weighted algorithm for computing pointwise mutual information. They added the weight to pointwise mutual information to reduce the bias of giving longer tweet large score. WAPMI of token t in class c can be computed by,

$$\text{WAPMI}(t,c) = \sum_{c \in C} \sum_{d \in D_c} \alpha_d p(t|d) \log_2 \frac{p(t,c)}{p(t)p(c)}$$

where d is the tweet that contains token t, D_c is the set of tweets in class c. $p(t|d)$ is the conditional probability of t, which is computed by,

$$p(t|d) = \frac{n(t,d)}{|d|}$$

where $n(t,d)$ is the frequency of token t occurring in tweet d and $|d|$ is the total size of tweet d.

α_d is the weight of token t. The authors gave 3 different weighting method in [21], which are:

- $\alpha_d = p(c) \times |d| / \sum_{d \in D_c} |d|$. Each tweet has been given a weight correlation to their length $|d|$.
- $\alpha_d = 1 / \sum_{c \in C} |c|$. This will give tweets in the same class an equal weight.
- $\alpha_d = 1/(|D_c| \times |C|)$. This will give equal weight to the classes by normalization by class size.

χ^2 is another feature selection method which can measure the relationship between the token and class. The lower the χ^2 score is, the token and class are more independent to each other. χ^2 can be computed by the deviation between the excepted frequency and the observed frequency of token t in class c.

$$\chi^2(t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

where e_t, e_c, $N_{e_t e_c}$ are same as the formula of MI.

4.4 Experiments

We first tried unigram and bigram model using two different Naive Bayes classification, Multinomial and Bernoulli models. All the tweets have already been split by regular expression. Then stop words have been removed and all the tokens have been normalized in the previous section. So the only thing we need to do to convert tweets into vectors is to generate unigrams and bigrams and count the frequency. For Multinomial model, each location in the vector is the frequency of the gram of this location; for Bernoulli model, each location in the vector is 1 if the gram occurring in the tweets of this user. The dimension of word vectors using unigram is 2,293,666 and the dimension of word vectors using bigram is 17,485,806.

All these processes have been done by C++ so that we can manually control the memory and achieve a better performance. When implementing, we used a feature-frequency dictionary in memory because the matrix of user vectors are so sparse. We tested Multinomial Naive Bayes classifier and Bernoulli Naive Bayes classifier on processed dataset. Table 14 and 15 show the result of 10-fold cross validation of these two classification models on unigram and bigram model. Fig. 5 and 6 show the variant of classifiers during 10 runs.

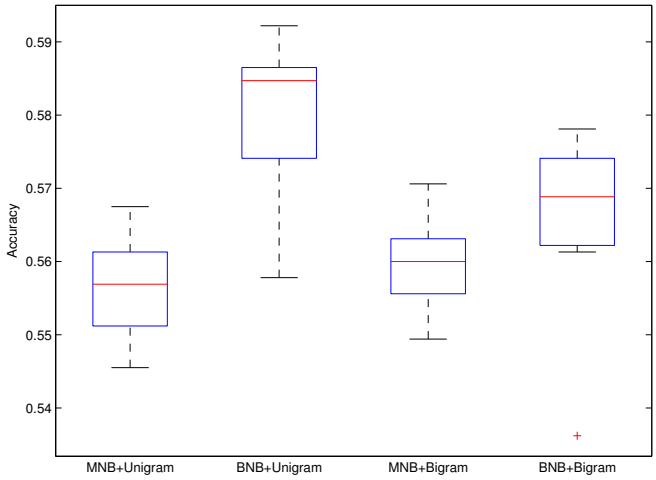

FIGURE 5: Accuracy for BNB and MNB. Both unigram and bigram models are tested.

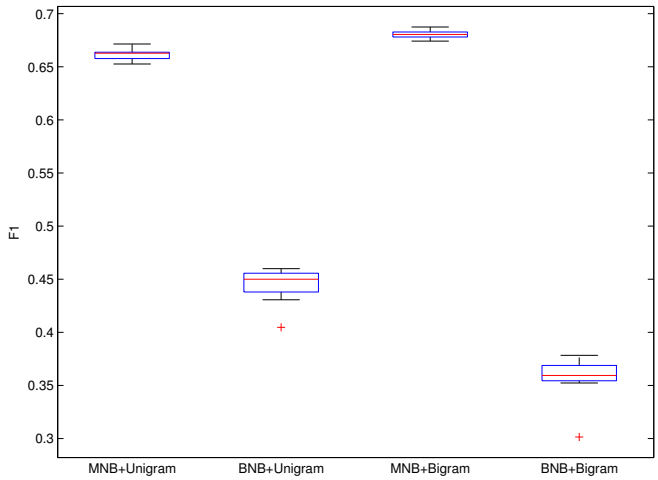

FIGURE 6: F1 for BNB and MNB. Both unigram and bigram models are tested.

The tables show that bigram model outperforms unigram model when using MNB classifier both on accuracy and F1. However the results of MNB are not good enough. This is because the probabilities of grams in both classes is similar. Fig. 7 shows the comparison of probabilities on each unigram feature in different classes. The red dots in this figure represent the probability of this unigram feature in suspended class is higher than it in non-suspended class and the blue dots represent the probability of this unigram feature in suspended class is lower than that in non-suspended class. The subplots in this plot indicate top 10^3, 10^4, 10^5 and all unigram features sorted by the probability. In top 10^3 features, the probabilities of 680 features in suspended class is higher. This number in top 10^4 is 5,952, in top 10^5 is 71,223 and in all the features is 1,318,360. This number is surprisingly high, resulting in if a normal user send a tweet in which all words are from top 10^5 unigram features, this user will be classified as suspended user on a extremely high probability.

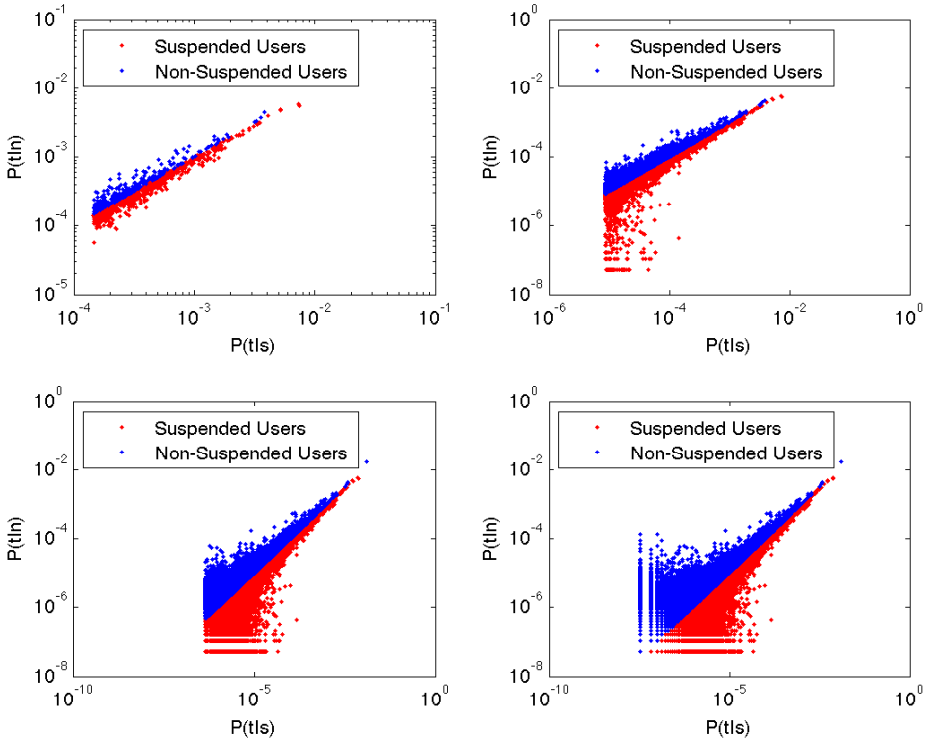

FIGURE 7: Probabilities of all unigram feature in both classes

Model	No.	TP	FN	FP	TN	Precision	Recall	Acc	F1
MNB	1	1002	132	849	285	54.13 %	88.36 %	56.75 %	67.14 %
MNB	2	986	148	851	283	53.67 %	86.95 %	55.95 %	66.37 %
MNB	3	996	138	876	258	53.21 %	87.83 %	55.29 %	66.27 %
MNB	4	978	156	845	289	53.65 %	86.24 %	55.86 %	66.15 %
MNB	5	961	172	851	282	53.04 %	84.82 %	54.85 %	65.26 %
MNB	6	975	158	872	261	52.79 %	86.05 %	54.55 %	65.44 %
MNB	7	985	148	846	287	53.80 %	86.94 %	56.13 %	66.46 %
MNB	8	967	166	818	315	54.17 %	85.35 %	56.58 %	66.28 %
MNB	9	992	141	867	266	53.36 %	87.56 %	55.52 %	66.31 %
MNB	10	977	156	861	272	53.16 %	86.23 %	55.12 %	65.77 %
MNB	Total	9819	1515	8536	2798	53.49 %	86.63 %	55.66 %	66.15 %
BNB	1	341	793	210	924	61.89 %	30.07 %	55.78 %	40.47 %
BNB	2	387	747	178	956	68.50 %	34.13 %	59.22 %	45.56 %
BNB	3	385	749	193	941	66.61 %	33.95 %	58.47 %	44.98 %
BNB	4	401	733	209	925	65.74 %	35.36 %	58.47 %	45.99 %
BNB	5	365	768	197	936	64.95 %	32.22 %	57.41 %	43.07 %
BNB	6	385	748	189	944	67.07 %	33.98 %	58.65 %	45.11 %
BNB	7	380	753	190	943	66.67 %	33.54 %	58.38 %	44.63 %
BNB	8	395	738	195	938	66.95 %	34.86 %	58.83 %	45.85 %
BNB	9	380	753	222	911	63.12 %	33.54 %	56.97 %	43.80 %
BNB	10	384	749	189	944	67.02 %	33.89 %	58.61 %	45.02 %
BNB	Total	3803	7531	1972	9362	65.85 %	33.55 %	58.08 %	44.46 %

TABLE 14: Unigram Classification Result

Model	No.	TP	FP	FN	TN	Precision	Recall	Acc	F1
MNB	1	1066	68	923	211	53.59 %	94.00 %	56.31 %	68.27 %
MNB	2	1070	64	924	210	53.66 %	94.36 %	56.44 %	68.41 %
MNB	3	1061	73	935	199	53.16 %	93.56 %	55.56 %	67.80 %
MNB	4	1066	68	936	198	53.25 %	94.00 %	55.73 %	67.98 %
MNB	5	1057	76	916	217	53.57 %	93.29 %	56.22 %	68.06 %
MNB	6	1056	77	944	189	52.80 %	93.20 %	54.94 %	67.41 %
MNB	7	1070	63	910	223	54.04 %	94.44 %	57.06 %	68.74 %
MNB	8	1064	69	921	212	53.60 %	93.91 %	56.31 %	68.25 %
MNB	9	1056	77	939	194	52.93 %	93.20 %	55.16 %	67.52 %
MNB	10	1065	68	934	199	53.28 %	94.00 %	55.78 %	68.01 %
MNB	Total	10631	703	9282	2052	53.39 %	93.80 %	55.95 %	68.04 %
BNB	1	227	907	145	989	61.02 %	20.02 %	53.62 %	30.15 %
BNB	2	281	853	109	1025	72.05 %	24.78 %	57.58 %	36.88 %
BNB	3	270	864	129	1005	67.67 %	23.81 %	56.22 %	35.23 %
BNB	4	285	849	117	1017	70.90 %	25.13 %	57.41 %	37.11 %
BNB	5	269	864	116	1017	69.87 %	23.74 %	56.75 %	35.44 %
BNB	6	277	856	118	1015	70.13 %	24.45 %	57.02 %	36.26 %
BNB	7	275	858	112	1021	71.06 %	24.27 %	57.19 %	36.18 %
BNB	8	272	861	124	1009	68.69 %	24.01 %	56.53 %	35.58 %
BNB	9	276	857	137	996	66.83 %	24.36 %	56.13 %	35.71 %
BNB	10	291	842	114	1019	71.85 %	25.68 %	57.81 %	37.84 %
BNB	Total	2723	8611	1221	10113	69.04 %	24.03 %	56.63 %	35.65 %

TABLE 15: Bigram Classification Result

And in both tables we can find out that Bernoulli model works really bad on unigram and bigram models. According to the testing formula of Bernoulli model,

$$c = \arg\max_{c \in C} P(c) \prod_{1 \le k \le |V|} P(t_k|c)^{x_i}(1 - P(t_k|c)^{1-x_i})$$

, the rare words are important parameters here. If the number of rare words in a class is significant higher than that in another class, because of each rare word that is not occurring in user's tweets will contribute a $1 - P(t)$ to the total value, the result of classifying will more likely to be the class with more rare words. For example, if user frequency of words in class A is $(3, 2, 2)$, while this frequency of words in class B is

$(2, 1, 1)$. Supposing the number of users in both dataset is 4. And the testing case only contains the first word in training dataset. We can compute the probabilities:

$$c_A = \log\frac{4}{8} + \log\frac{3}{4} + \log(1 - \frac{2}{4}) + \log(1 - \frac{2}{4}) \qquad = -1.67$$
$$c_B = \log\frac{4}{8} + \log\frac{2}{4} + \log(1 - \frac{1}{4}) + \log(1 - \frac{1}{4}) \qquad = -1.27$$

So the testing case will be labelled as class B. In our dataset, the number of rare words in non-suspended user dataset is much smaller than that of suspended user dataset. To be more precisely, the total user frequency of words of which the user frequency is less than 5 in suspended user dataset is 1,638,942 while that number in non-suspended user dataset is 1,195,925, which is the reason why BNB classifier tends to classify users into non-suspended user.

4.4.1 Feature Selection

We also performed MI, PMI, WAPMI and χ^2 feature selection methods on the N-Gram models. In order to analyse the relationship between the size of selected feature set and the performance of classification, We run 10-fold cross validation on the whole dataset. We first divided the whole dataset into 10 subdatasets equally and for each running of validation, 9 of 10 sub datasets has been merged as training dataset and the rest one has been used as testing dataset. The training dataset will be split into tokens and the tokens will be normalized. We then counted N_{11}, N_{10}, N_{01} and N_{00} for each unigram and bigram generated based on the tokens of training dataset.

We first tried MI based on the count of users who sent tweets containing the grams. We found that the result of MI is not good because it tends to select the features from suspended users rather than non-suspended users. In order to solve the problem that the feature selection method will have more probability to select the features from suspended users than select from non-suspended users, we used token frequency instead of number of users. We changed the definitions of N_{11} and N_{10} to the total frequency of this feature occurring in the whole suspended user dataset and non-suspended user dataset. We then run experiments based on the token frequency instead of user frequency and computed the score for each feature. The formulas of these feature selection methods listed below,

$$\mathrm{MI}(t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{N_{e_t e_c}}{N} \log_2 \frac{N_{e_t e_c}}{E_{e_t e_c}}$$

$$\mathrm{PMI}(t,c) = \sum_{e_c \in \{0,1\}} |\log_2 \frac{N_{1 e_c}}{E_{1 e_c}}|$$

$$\mathrm{WAPMI}(t,c) = \sum_{c \in C} \sum_{d \in D_c} \alpha_d p(t|d) \log_2 \frac{p(t,c)}{p(t)p(c)}$$

$$\chi^2(t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

where α_d in WAPMI is defined as $p(c) \times |d| / \sum_{d \in D_c} |d|$.

Table 16 to 23 show the details of results of these 4 feature selection algorithms on both unigram and bigram models. In these tables, STF represents normalized token frequency in suspended user dataset, NTF represents token frequency in non-suspended user dataset, SUF represents user frequency in suspended uses dataset and NUF represents user frequency in non-suspended user dataset. We normalized the token frequency in suspended user dataset by,

$$STF = t_s \frac{T_{suspended}}{T_{non-suspended}}$$

where t_s is the token frequency in suspended user dataset, $T_{suspended}$ and $T_{non-suspended}$ are the total token frequency in suspended and non-suspended user datasets. We highlighted some distinguished grams in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	rt	239386.37	318635	4499	3340	2.10×10^{-4}
2	flight	2064.22	10172	1252	856	1.13×10^{-4}
3	#fb	2497.10	10961	341	283	1.10×10^{-4}
4	**lol**	141102.19	104816	4958	2931	0.97×10^{-4}
5	uk	2600.99	9561	906	590	0.81×10^{-4}
6	htm	149.03	3292	142	99	0.71×10^{-4}
7	http://uk	0.00	2494	0	1	0.70×10^{-4}
8	prodotti	0.62	2467	1	1	0.69×10^{-4}
9	nuovi	1.86	2473	2	2	0.69×10^{-4}
10	**snarf**	2727.76	8	14	7	0.61×10^{-4}
11	gue	2618.31	8212	155	135	0.58×10^{-4}
12	cheap	1387.07	5804	1065	672	0.56×10^{-4}
13	blog	13046.39	23263	2105	1424	0.54×10^{-4}
14	eu	29941.07	18141	537	436	0.52×10^{-4}
15	cc	672.82	4048	301	246	0.52×10^{-4}
16	love	137609.47	111429	7411	5006	0.50×10^{-4}
17	@cutie_jessica05	0.00	1674	0	1	0.47×10^{-4}
18	**f*** **	27380.28	16684	2876	1476	0.47×10^{-4}
19	@mike_tizzal	0.62	1661	1	2	0.47×10^{-4}
20	ga	4498.86	10485	634	490	0.46×10^{-4}
21	#gagavmas	2.47	1657	2	2	0.46×10^{-4}
22	**@nike**	1980.12	0	3	0	0.45×10^{-4}
23	fifty8	0.00	1589	0	1	0.45×10^{-4}
24	en	16877.37	27085	1018	875	0.44×10^{-4}
25	el	15212.64	24360	1080	788	0.39×10^{-4}
26	gw	3782.75	8830	132	177	0.39×10^{-4}
27	lo	9562.94	16923	1057	762	0.38×10^{-4}
28	#funsat	0.62	1323	1	3	0.37×10^{-4}
29	@dolphinnancy	0.00	1292	0	1	0.36×10^{-4}
30	admartindia	0.00	1262	0	1	0.35×10^{-4}

TABLE 16: Top 30 Unigram Features by MI. Terms in bold fonts are distinguished ones in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	http://uk	0.00	2494	0	1	11.98
2	@cutie_jessica05	0.00	1674	0	1	11.40
3	fifty8	0.00	1589	0	1	11.33
4	@dolphinnancy	0.00	1292	0	1	11.03
5	admartindia	0.00	1262	0	1	11.00
6	@novawildstar	0.00	1261	0	1	10.99
7	prodotti	0.62	2467	1	1	10.96
8	geeksroom	0.00	1229	0	1	10.96
9	**@nike**	1980.12	0	3	0	10.95
10	http://tinyurl.com/czzu6n	0.00	1199	0	1	10.92
11	@mzsreyes	0.00	1098	0	1	10.80
12	http://wp.me/pivby	0.00	1036	0	1	10.71
13	@viramutiara	0.00	1015	0	1	10.68
14	@willamutiara	0.00	950	0	1	10.59
15	@atirahn	0.00	947	0	1	10.58
16	@dmand21	0.00	917	0	1	10.54
17	@dirtymink	0.00	888	0	1	10.49
18	@ifew	0.00	879	0	2	10.47
19	@coolzebras	0.00	857	0	1	10.44
20	@kay_dead	0.00	851	0	1	10.43
21	@mike_tizzal	0.62	1661	1	2	10.39
22	tuwallstreet	0.00	807	0	1	10.35
23	@capr1cemd	0.00	801	0	1	10.34
24	@panda_baggins	0.00	784	0	1	10.31
25	@babimalez	0.00	757	0	1	10.26
26	@stacygardell	0.00	755	0	1	10.26
27	@shintashasya	0.00	744	0	1	10.23
28	@cher666	0.00	732	0	1	10.21
29	@xaveriouseja	0.00	714	0	1	10.18
30	@kathyj3490	0.00	708	0	1	10.16

TABLE 17: Top 30 Unigram Features by PMI. Terms in bold fonts are distinguished ones in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	**f*****	27380.28	16684	2876	1476	4.46×10^{-5}
2	**lol**	141102.19	104816	4958	2931	4.38×10^{-5}
3	eu	29941.07	18141	537	436	3.72×10^{-5}
4	**@nike**	1980.12	0	3	0	3.57×10^{-5}
5	**s*****	22087.39	14140	2970	1622	2.96×10^{-5}
6	yg	4487.11	9175	124	201	2.82×10^{-5}
7	**snarf**	2727.76	8	14	7	2.77×10^{-5}
8	http://uk	0.00	2494	0	1	2.76×10^{-5}
9	**http://blip.fm/**	7436.88	3381	242	144	2.65×10^{-5}
10	gw	3782.75	8830	132	177	2.31×10^{-5}
11	**lmao**	21995.87	13854	1714	813	2.31×10^{-5}
12	**b****	10994.53	6362	2465	1255	2.28×10^{-5}
13	dont	25473.13	17505	3606	2064	2.20×10^{-5}
14	dnt	4255.21	1638	631	307	2.18×10^{-5}
15	**#teamdemi**	936.88	0	7	0	2.07×10^{-5}
16	smh	5398.63	2590	505	222	1.91×10^{-5}
17	qe	2735.80	635	143	76	1.90×10^{-5}
18	@cutie_jessica05	0.00	1674	0	1	1.85×10^{-5}
19	ich	4193.99	1679	206	122	1.76×10^{-5}
20	fifty8	0.00	1589	0	1	1.76×10^{-5}
21	#nowplaying	10076.21	6129	1222	622	1.64×10^{-5}
22	photo	16514.99	11427	2298	1423	1.63×10^{-5}
23	aja	2508.85	6197	146	189	1.54×10^{-5}
24	@parisfilmes	851.54	0	3	0	1.54×10^{-5}
25	wed	7172.82	4153	1949	1223	1.53×10^{-5}
26	quiz	6138.24	3359	1320	631	1.51×10^{-5}
27	#funsat	0.62	1323	1	3	1.46×10^{-5}
28	**a****	14032.12	9251	2845	1526	1.45×10^{-5}
29	pra	14516.94	10164	419	310	1.45×10^{-5}
30	love	137609.47	111429	7411	5006	1.44×10^{-5}

TABLE 18: Top 30 Unigram Features by WAPMI. Terms in bold fonts are distinguished ones in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	rt	239386.37	318635	4499	3340	14600.02
2	flight	2064.22	10172	1252	856	7870.01
3	#fb	2497.10	10961	341	283	7725.44
4	**lol**	141102.19	104816	4958	2931	6437.60
5	uk	2600.99	9561	906	590	5691.37
6	htm	149.03	3292	142	99	4520.19
7	gue	2618.31	8212	155	135	4066.28
8	http://uk	0.00	2494	0	1	4031.19
9	nuovi	1.86	2473	2	2	3986.41
10	prodotti	0.62	2467	1	1	3983.92
11	cheap	1387.07	5804	1065	672	3920.81
12	blog	13046.39	23263	2105	1424	3808.54
13	cc	672.82	4048	301	246	3586.53
14	eu	29941.07	18141	537	436	3387.25
15	love	137609.47	111429	7411	5006	3341.52
16	ga	4498.86	10485	634	490	3263.96
17	en	16877.37	27085	1018	875	3102.14
18	**f*****	27380.28	16684	2876	1476	3038.40
19	el	15212.64	24360	1080	788	2766.51
20	gw	3782.75	8830	132	177	2756.93
21	lo	9562.94	16923	1057	762	2706.62
22	**snarf**	2727.76	8	14	7	2705.14
23	@cutie_jessica05	0.00	1674	0	1	2705.08
24	@mike_tizzal	0.62	1661	1	2	2680.45
25	#gagavmas	2.47	1657	2	2	2663.18
26	fifty8	0.00	1589	0	1	2567.62
27	di	6167.30	11822	725	603	2373.58
28	**lmao**	21995.87	13854	1714	813	2170.84
29	yg	4487.11	9175	124	201	2163.64
30	aja	2508.85	6197	146	189	2145.61

TABLE 19: Top 30 Unigram Features by χ^2. Terms in bold fonts are distinguished ones in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	film blog	4.95	3039	5	3	9.75×10^{-5}
2	cheap flight	21.03	2685	27	16	8.20×10^{-5}
3	flight cc	0.00	2494	0	1	8.13×10^{-5}
4	nuovi prodotti	0.00	2467	0	1	8.04×10^{-5}
5	uk flight	0.62	1928	1	1	6.26×10^{-5}
6	**www paid**	1952.29	0	1	0	5.24×10^{-5}
7	**paid draw**	1952.91	1	2	1	5.22×10^{-5}
8	fifty8 uk	0.00	1514	0	1	4.93×10^{-5}
9	www fifty8	0.00	1509	0	1	4.91×10^{-5}
10	#gagavmas #gagavmas	0.00	1507	0	1	4.90×10^{-5}
11	air flight	3.09	1523	5	4	4.86×10^{-5}
12	http://uk cheap	0.00	1459	0	1	4.75×10^{-5}
13	updat blog	2075.97	116	144	76	4.23×10^{-5}
14	2009 price	0.62	1280	1	1	4.14×10^{-5}
15	book cheap	0.62	1250	1	6	4.04×10^{-5}
16	blog post	2451.96	6393	463	400	3.89×10^{-5}
17	**snarf snarf**	1359.24	1	4	1	3.62×10^{-5}
18	releas sep	0.00	1096	0	1	3.56×10^{-5}
19	cordless show	0.62	986	1	1	3.18×10^{-5}
20	cheap uk	0.00	926	0	1	3.01×10^{-5}
21	#stay #stay	1159.50	7	3	1	2.98×10^{-5}
22	rt @kapanlagicom	1040.15	3	3	2	2.72×10^{-5}
23	cc cheap	0.00	816	0	1	2.65×10^{-5}
24	flight uk	0.00	815	0	2	2.64×10^{-5}
25	#veronicamars #veronicamars	5.57	857	1	1	2.64×10^{-5}
26	vote ak	0.00	797	0	1	2.58×10^{-5}
27	http://uk air	0.00	763	0	1	2.47×10^{-5}
28	cheap air	1.86	760	3	4	2.41×10^{-5}
29	**result credit**	892.97	0	1	0	2.38×10^{-5}
30	99 descript	0.00	726	0	1	2.35×10^{-5}

TABLE 20: Top 30 Bigram Features by MI. Terms in bold fonts are distinguished ones in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	flight cc	0.00	2494	0	1	11.96
2	nuovi prodotti	0.00	2467	0	1	11.94
3	fifty8 uk	0.00	1514	0	1	11.24
4	www fifty8	0.00	1509	0	1	11.24
5	#gagavmas #gagavmas	0.00	1507	0	1	11.23
6	http://uk cheap	0.00	1459	0	1	11.19
7	**www paid**	1952.29	0	1	0	10.95
8	releas sep	0.00	1096	0	1	10.77
9	uk flight	0.62	1928	1	1	10.59
10	cheap uk	0.00	926	0	1	10.53
11	cc cheap	0.00	816	0	1	10.35
12	flight uk	0.00	815	0	2	10.35
13	vote ak	0.00	797	0	1	10.31
14	http://uk air	0.00	763	0	1	10.25
15	99 descript	0.00	726	0	1	10.18
16	scade il	0.00	677	0	1	10.08
17	@www amazon	0.00	655	0	1	10.03
18	www cordless	0.00	641	0	1	10.00
19	2009 price	0.62	1280	1	1	10.00
20	book cheap	0.62	1250	1	6	9.96
21	**paid draw**	1952.91	1	2	1	9.95
22	rt @atirahn	0.00	615	0	1	9.94
23	neue da	0.00	613	0	1	9.94
24	cc book	0.00	604	0	1	9.92
25	click http://bit.ly/okvd7	0.00	600	0	1	9.91
26	cc fli	0.00	593	0	1	9.89
27	matter click	0.00	593	0	2	9.89
28	tuwallstreet comment	0.00	583	0	1	9.86
29	**result credit**	892.97	0	1	0	9.82
30	ak perform	0.00	558	0	1	9.80

TABLE 21: Top 30 Bigram Features by PMI. Terms in bold fonts are distinguished ones in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	flight cc	0.00	2494	0	1	3.19×10^{-5}
2	nuovi prodotti	0.00	2467	0	1	3.16×10^{-5}
3	**www paid**	1952.29	0	1	0	2.53×10^{-5}
4	book cheap	0.62	1250	1	6	2.46×10^{-5}
5	updat blog	2075.97	116	144	76	2.31×10^{-5}
6	**snarf snarf**	1359.24	1	4	1	2.18×10^{-5}
7	fifty8 uk	0.00	1514	0	1	1.94×10^{-5}
8	www fifty8	0.00	1509	0	1	1.93×10^{-5}
9	#gagavmas #gagavmas	0.00	1507	0	1	1.93×10^{-5}
10	http://uk cheap	0.00	1459	0	1	1.87×10^{-5}
11	**super kit**	828.04	0	3	0	1.75×10^{-5}
12	**paid draw**	1952.91	1	2	1	1.60×10^{-5}
13	**#teamdemi #teamdemi**	843.50	0	2	0	1.54×10^{-5}
14	rt @parisfilmes	831.75	0	2	0	1.52×10^{-5}
15	kit lua	827.42	0	2	0	1.51×10^{-5}
16	#stay #stay	1159.50	7	3	1	1.49×10^{-5}
17	flight uk	0.00	815	0	2	1.47×10^{-5}
18	releas sep	0.00	1096	0	1	1.40×10^{-5}
19	cheap uk	0.00	926	0	1	1.19×10^{-5}
20	result credit	892.97	0	1	0	1.16×10^{-5}
21	song wow	548.52	8	35	8	1.13×10^{-5}
22	@parisfilmes @ucicinemas	842.26	0	1	0	1.09×10^{-5}
23	@espacozoficial @andreianapoleao	828.66	0	1	0	1.07×10^{-5}
24	@ucicinemas @espacozoficial	828.04	0	1	0	1.07×10^{-5}
25	@eric_twittando http://migre.me/baje	826.80	0	1	0	1.07×10^{-5}
26	matter click	0.00	593	0	2	1.07×10^{-5}
27	cc cheap	0.00	816	0	1	1.04×10^{-5}
28	#glee #glee	484.21	6	16	3	1.03×10^{-5}
29	#australiawantsjonas #australiawantsjonas	483.59	0	3	0	1.02×10^{-5}
30	vote ak	0.00	797	0	1	1.02×10^{-5}

TABLE 22: Top 30 Bigram Features by WAPMI. Terms in bold fonts are distinguished ones in suspended class.

No.	Feature	STF	NTF	SUF	NUF	Score
1	film blog	4.95	3039	5	3	4820.73
2	cheap flight	21.03	2685	27	16	4163.86
3	flight cc	0.00	2494	0	1	3979.32
4	nuovi prodotti	0.00	2467	0	1	3936.21
5	uk flight	0.62	1928	1	1	3072.15
6	fifty8 uk	0.00	1514	0	1	2414.83
7	air flight	3.09	1523	5	4	2411.31
8	www fifty8	0.00	1509	0	1	2406.85
9	#gagavmas #gagavmas	0.00	1507	0	1	2403.66
10	blog post	2451.96	6393	463	400	2355.50
11	http://uk cheap	0.00	1459	0	1	2327.03
12	2009 price	0.62	1280	1	1	2037.70
13	book cheap	0.62	1250	1	6	1989.81
14	**www paid**	1952.29	0	1	0	1975.89
15	**paid draw**	1952.91	1	2	1	1973.90
16	updat blog	2075.97	116	144	76	1813.15
17	releas sep	0.00	1096	0	1	1747.55
18	cordless show	0.62	986	1	1	1568.37
19	cheap uk	0.00	926	0	1	1476.18
20	**snarf snarf**	1359.24	1	4	1	1372.42
21	#veronicamars #veronicamars	5.57	857	1	1	1334.14
22	cc cheap	0.00	816	0	1	1300.58
23	flight uk	0.00	815	0	2	1298.98
24	vote ak	0.00	797	0	1	1270.25
25	http://uk air	0.00	763	0	1	1215.98
26	cheap air	1.86	760	3	4	1200.48
27	99 descript	0.00	726	0	1	1156.91
28	#stay #stay	1159.50	7	3	1	1154.43
29	#charice pyramid	0.62	719	1	1	1142.16
30	scade il	0.00	677	0	1	1078.69

TABLE 23: Top 30 Bigram Features by χ^2. Terms in bold fonts are distinguished ones in suspended class.

In these tables, we can figure out that MI and χ^2 tend to select popular words while PMI and WAPMI tend to select rare words. And many of selected features show some characters about why users were suspended. Unigrams selected by MI and χ^2, such as "uk", "htm", "cc" and "en", are more likely to be appeared in a URL, which shows that URL plays an important role in distinguishing between suspended users and non-suspended users. Although the token frequencies of "uk" or "htm" in suspended users are larger than these words in non-suspended users, the user frequencies of these words are bigger in suspended user dataset, which means that more suspended users are likely to send URL. Some studies [2, 22] also indicated that spammers will try to use URL as text instead of shorten URL. In our splitting and normalizing step, we used regular expression to split shorten URL into one token. So the reason why we now got some part of URL in tokens is that these users tried to send URL as text rather than shorten URL to avoid being blocked by URL blacklist.

The selected bigrams gave more evidences to this assumption. The bigram feature, "www paid", selected by WAPMI was only sent by a spammer who has already been suspended. Before suspended by Twitter, this user sent 3,157 tweets and all of them contains "www.Paid-To-Draw.com", which is a website that can make money by drawing. Tweets such as "Do you want to recieve huge notoriety as an artist? Visit www.Paid-To-Draw.com", "Working on promoting a fantastic product www.Paid-To-Draw.com", include the URL as text in order to avoid blocking by shorten URL detection. Another example is that inside non-suspended users, there is also a user who sending URL as text to avoid being blocked. The bigram feature, "www fifty8", was sent by a user who has not been suspended but keeping sending spams. This user successfully passed URL blacklist detection by using this method.

Another main reason of suspending is abusive activities. In the unigram features, we can find there is a large number of suspended users who sent tweets include f-word, b-word or s-word. The total number of suspended users who sent these words is 4,025 and this number of non-suspended users is 2,333. And these suspended users sent 24 tweets containing these words on average and these non-suspended users only sent 16 tweets. We will discuss this later in experiments on bad words.

Sending massive mentions and hashtags is another reason of suspending. In bigram features selected by WAPMI (Table 22), we can found features ranked from 16 to 20, which contain a large number of mentions and hashtags. In fact all of these features are extract from the tweets sent by same user. This suspended user sent a lot of

tweets containing mentions and hashtags to get attention from other users.

The features selected by MI, PMI, WAPMI are slightly different from each other. MI and χ^2 tend to select popular words and resulting in the similar results. Fig. 8 shows the differences between these feature selection methods. Each plot contains top 10,000 features selected by different methods. Each feature is located by token frequency in suspended user dataset and non-suspended user dataset. Compared to MI and χ^2, which selected the popular features, PMI tends to select those rare features, especially those features which occur only in one dataset. And because of the normalization factor α_d, WAPMI selected both rare features and popular features.

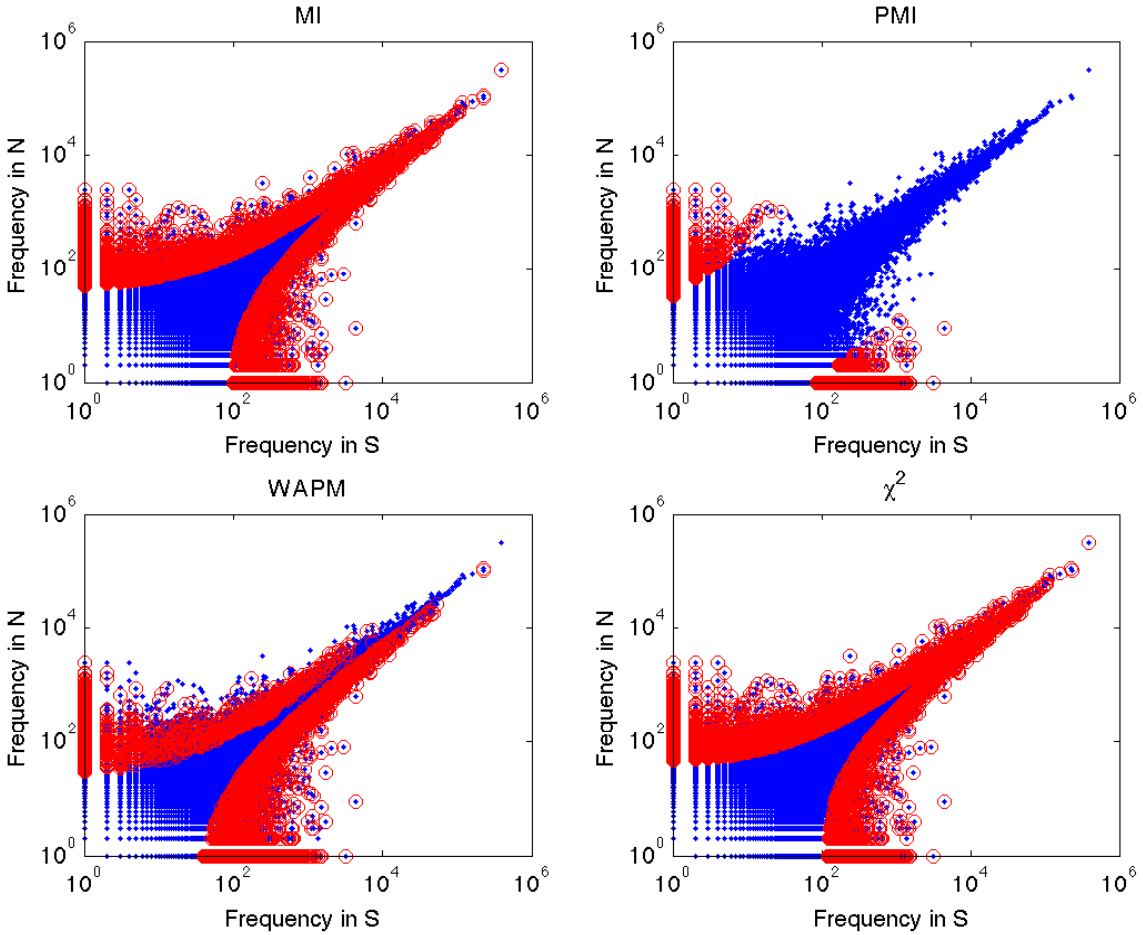

FIGURE 8: Locations of Features selected by MI, PMI, WAPMI and χ^2

In order to evaluate the performances of each feature selection methods, we tried to adjust the size of features to filter the dataset and then run 10-fold cross validation on classification by using Multinomial Naive Bayes classifier. The relationship between the size of top selected features and the result of classification shows in Table 24 and

25.

In Fig 9, both the accuracy and F1 have increased , especially on the results of PMI and WAPMI. The accuracy and F1 on unigram now can be 67.70% and 68.98% by using top 10^6 features selected by PMI; and the accuracy and F1 on bigram now can be 76.75% and 78.54% by using top 1.5×10^7 features selected by WAPMI.

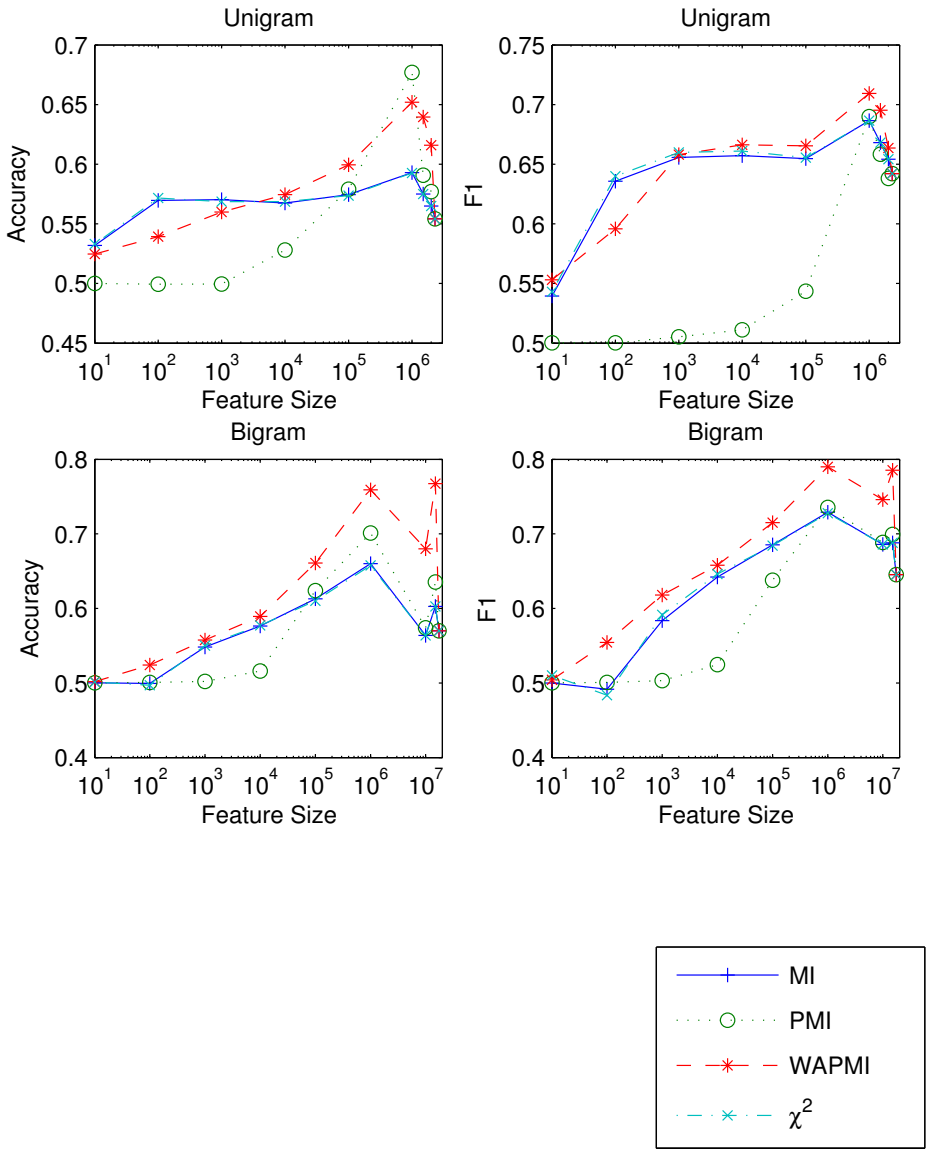

FIGURE 9: Feature Size vs Accuracy and F1

Model	Feature Size	Precision	Recall	Accuracy	F1
Unigram + MI	10	53.08 %	54.83 %	53.19 %	53.94 %
Unigram + MI	100	55.11 %	75.10 %	56.97 %	63.57 %
Unigram + MI	1000	54.70 %	81.84 %	57.03 %	65.57 %
Unigram + MI	10000	54.42 %	82.93 %	56.74 %	65.72 %
Unigram + MI	100000	55.08 %	80.67 %	57.44 %	65.46 %
Unigram + MI	1000000	55.82 %	89.17 %	59.29 %	68.65 %
Unigram + MI	1500000	54.81 %	85.52 %	57.50 %	66.80 %
Unigram + MI	2000000	54.28 %	82.30 %	56.49 %	65.42 %
Unigram + PMI	10	50.01 %	50.04 %	50.01 %	50.03 %
Unigram + PMI	100	49.93 %	50.11 %	49.93 %	50.02 %
Unigram + PMI	1000	49.96 %	51.13 %	49.96 %	50.54 %
Unigram + PMI	10000	53.02 %	49.35 %	52.81 %	51.12 %
Unigram + PMI	100000	59.35 %	50.14 %	57.90 %	54.36 %
Unigram + PMI	1000000	66.35 %	71.84 %	67.70 %	68.98 %
Unigram + PMI	1500000	56.50 %	78.86 %	59.07 %	65.83 %
Unigram + PMI	2000000	55.76 %	74.55 %	57.70 %	63.80 %
Unigram + WAPMI	10	52.18 %	58.82 %	52.46 %	55.30 %
Unigram + WAPMI	100	53.07 %	67.94 %	53.93 %	59.59 %
Unigram + WAPMI	1000	53.81 %	84.70 %	56.00 %	65.81 %
Unigram + WAPMI	10000	54.81 %	84.92 %	57.45 %	66.62 %
Unigram + WAPMI	100000	57.14 %	79.63 %	59.95 %	66.54 %
Unigram + WAPMI	1000000	60.89 %	84.97 %	65.20 %	70.94 %
Unigram + WAPMI	1500000	60.22 %	82.21 %	63.95 %	69.52 %
Unigram + WAPMI	2000000	59.03 %	75.77 %	61.59 %	66.36 %
Unigram + χ^2	10	53.18 %	55.46 %	53.31 %	54.29 %
Unigram + χ^2	100	55.20 %	76.12 %	57.17 %	63.99 %
Unigram + χ^2	1000	54.47 %	83.70 %	56.86 %	65.99 %
Unigram + χ^2	10000	54.45 %	84.10 %	56.87 %	66.10 %
Unigram + χ^2	100000	55.00 %	81.15 %	57.37 %	65.56 %
Unigram + χ^2	1000000	55.79 %	89.21 %	59.26 %	68.65 %
Unigram + χ^2	1500000	54.81 %	85.52 %	57.50 %	66.80 %
Unigram + χ^2	2000000	54.28 %	82.31 %	56.49 %	65.42 %

TABLE 24: Classification Results of Unigram on Different Feature Sizes

Model	Feature Size	Precision	Recall	Accuracy	F1
Bigram + MI	10	50.01 %	49.96 %	50.01 %	49.99 %
Bigram + MI	100	49.93 %	48.44 %	49.93 %	49.17 %
Bigram + MI	1000	54.12 %	63.31 %	54.82 %	58.35 %
Bigram + MI	10000	55.58 %	76.01 %	57.63 %	64.21 %
Bigram + MI	100000	57.74 %	84.25 %	61.29 %	68.52 %
Bigram + MI	1000000	60.61 %	91.48 %	66.01 %	72.91 %
Bigram + MI	10000000	53.58 %	95.24 %	56.37 %	68.58 %
Bigram + MI	15000000	56.64 %	87.63 %	60.28 %	68.81 %
Bigram + PMI	10	50.01 %	50.02 %	50.01 %	50.02 %
Bigram + PMI	100	50.08 %	50.09 %	50.08 %	50.09 %
Bigram + PMI	1000	50.20 %	50.43 %	50.20 %	50.31 %
Bigram + PMI	10000	51.55 %	53.41 %	51.61 %	52.46 %
Bigram + PMI	100000	61.49 %	66.26 %	62.38 %	63.78 %
Bigram + PMI	1000000	65.98 %	83.08 %	70.13 %	73.55 %
Bigram + PMI	10000000	54.27 %	94.18 %	57.41 %	68.86 %
Bigram + PMI	15000000	59.51 %	84.75 %	63.55 %	69.93 %
Bigram + WAPMI	10	50.18 %	50.67 %	50.19 %	50.43 %
Bigram + WAPMI	100	52.13 %	59.22 %	52.42 %	55.45 %
Bigram + WAPMI	1000	54.38 %	71.57 %	55.76 %	61.80 %
Bigram + WAPMI	10000	56.36 %	79.03 %	58.92 %	65.80 %
Bigram + WAPMI	100000	61.66 %	85.12 %	66.10 %	71.52 %
Bigram + WAPMI	1000000	70.04 %	90.61 %	75.92 %	79.01 %
Bigram + WAPMI	10000000	61.85 %	93.98 %	68.01 %	74.60 %
Bigram + WAPMI	15000000	72.92 %	85.09 %	76.75 %	78.54 %
Bigram + χ^2	10	50.12 %	51.93 %	50.13 %	51.01 %
Bigram + χ^2	100	49.67 %	47.12 %	49.69 %	48.36 %
Bigram + χ^2	1000	54.16 %	65.16 %	55.01 %	59.15 %
Bigram + χ^2	10000	55.56 %	77.18 %	57.72 %	64.61 %
Bigram + χ^2	100000	57.49 %	84.47 %	61.01 %	68.42 %
Bigram + χ^2	1000000	60.42 %	91.50 %	65.78 %	72.78 %
Bigram + χ^2	10000000	53.58 %	95.24 %	56.37 %	68.58 %
Bigram + χ^2	15000000	56.64 %	87.63 %	60.28 %	68.81 %

TABLE 25: Classification Results of Bigram on Different Feature Sizes

4.4.2 Conclusion

We experimented classification using n-gram model with BNB and MNB. Feature selection methods, MI, PMI, WAPMI and χ^2 have also been used in these experiments. Among these methods, MNB + bigram + WAPMI is the best choice, which achieved 75.92 % accuracy and 79.01% F1. We compared BNB and MNB with full features and found both of them work badly. MNB tends to classify all users into suspended because suspended user dataset contain more top features than non-suspended user dataset, whil BNB tends to classify all users into non-suspended because non-suspended user dataset contain more rare features. Among the 4 feature selection methods, we found that MI and χ^2 tend to select popular words while PMI selects rare words and WAPMI select both popular and rare words. And the results of experiments show that rare words can increase the accuracy and F1, but there is a limitation because rare words may not occur in users' tweets. So in all methods, PMI, which select both popular words and rare words, performs best.

CHAPTER 5

Classification using word2vec and doc2vec

To apply other classification methods, we need to transform high dimensional representation of documents into shorter vectors. It is based on word embedding method that maps the tweets into low dimension vectors. We experimented with two methods: one uses the sum of word vectors, where each word vector is obtained from word2vec [23]. The other uses doc2vec introduced in [24].

5.1 Experiments of word2vec

In 2013, T. Mikolov et al. [23] proposed a new model named word2vec for computing continuous vector representations of words that cannot only map the words into lower dimension vector, and it can also represent the similarity of words by cosine distance between the vectors of two words. This model can be used in many areas in NLP, such as word clustering [25] and named entity recognition [26, 27]. It can even show the inside relationship between words by simple algebraic operations, for example that $vector(\text{"}King\text{"}) - vector(\text{"}Man\text{"}) + vector(\text{"}Woman\text{"})$ results in a vector that is closest to the vector representation of the word "$Queen$" [28].

Word2vec is based on distributed representations of words learnt by neural network. The task of word2vec is to train a matrix W, where each column represents a mapped word w. The goal of word2vec model is to maximize the accuracy of prediction of next word in sentence using the other word vectors in this sentence. More formally, Given a sequence of words, $w_1, w_2, ..., w_n$, word2vec model will try to maximize the average log probability

$$\frac{1}{n}\sum_{t=k}^{n-k}\log p(w_t|w_{t-k},...,w_{t+k})$$

By using CBOW (Continuous Bag-of Words Model) or Skip-gram model, it can easily process 783 million words within one day, which is a significant improvement on the performance. This makes it possible to train more complex model on much larger dataset, therefore the trained model can outperform the previous simple models.

However, word2vec can only map words to vectors, which is not enough for our classification. We need a mapping function that can convert the tweets into vectors. It can easily to come out with a naive idea that we can sum up all the word vectors in one tweet and the result vector can represent this tweet [29]. The following formula shows this idea:

$$vector(u) = \sum_{k=1}^{n} vector(w_k)$$

where w_k are the words in the tweets sent by user. This approach have some advantages. First is that it is easy to compute and understand. The previous example, $vector(\text{"}King\text{"}) + vector(\text{"}Woman\text{"}) \approx vector(\text{"}Queen\text{"}) + vector(\text{"}Man\text{"})$ illustrates that it is possible to sum vectors together without losing the meaning of sentences, and further more can somehow prove that this approach can be used to generate the vector of tweet. Second is that the dimension size of the result vector will still be the same as the dimension size of word vectors. Thus the tweets with different lengths can be easily converted into the same dimension vectors and then be used in classification.

But this simple method still have some problems in it. When converting words vector into tweet vector, we may face the problem that the word cannot be found in the vocabulary of pre-trained model. We have two options here to deal with this problem, one solution is to ignore this word, treating it like a all-zero vector; another is that we can assign a random vector to word and save this word together with the new vector into model, so that next time when we find the same word we can still use this vector. In our experiments, we tested all of these combinations.

In word2vec experiments, we tried to train 3 different dimensions: 300, 600, 1,000 using our tweets dataset with CBOW model. The parameters of training are window = 5 (indicating how many words will be used around current words when training vectors); min count = 5 (indicating the words of which the frequency is less than

5 will be removed). We also tried a pre-trained word vectors that is trained based on part of Google News dataset (about 100 billion words) [30]. This model contains 300-dimensional vectors for 3 million words.

After training, we used 4 different methods to concentrate word vectors to generate vectors for users. These 4 methods are depending on whether we need to normalize the concentrated user vectors and whether we need to discarding the missing words. If using normalization, the concentrated user vectors will be

$$vector(u) = \frac{\sum_{k=1}^{n} vector(w_k)}{n}$$

And if using discarding missing words method, the words which are not in the pre-trained word2vec model will be discarded. If we don't use discarding missing words method, the missing words will be randomly assigned word vectors.

After concentrating user vectors, we then used several classifiers to test which one performs best. The classifiers we used are listed in 29 and all the parameters remain default. Fig. 10 and 11 show the results of all the experiments. In these tables, $NORM = 1$ represents using normalization method, $DIS = 1$ represents using discarding missing words method and $DIS = 0$ represents assigning a random vector to missing word method.

We achieved the best accuracy (63.02 %) when using SVM with RBF function kernel on 1,000 dimension word vector model with normalization and discarding missing words and the best F1 (67.24 %) when using SVM with linear function kernel on 600 dimension word vector model with discarding missing words.

In these figures, we can conclude that first is the model we trained based on tweets can significantly improve the results of classification compared with pre-train model based on google news. This is because our model is focusing tweets area. Words and the way of using words may be very different between different area, so the trained word vectors can be very different.

Secondly, normalization will lose some important information on user vectors, resulting in bad results of classification. This is shown in bar plots of F1. The average F1 of different classifiers on normalized user vectors is 52.16 % while that on user vectors which are not using normalization is 63.29 %.

Thirdly, different dimensions don't play an important role in classification, which means that 300-dimension word vectors are good enough for classifying.

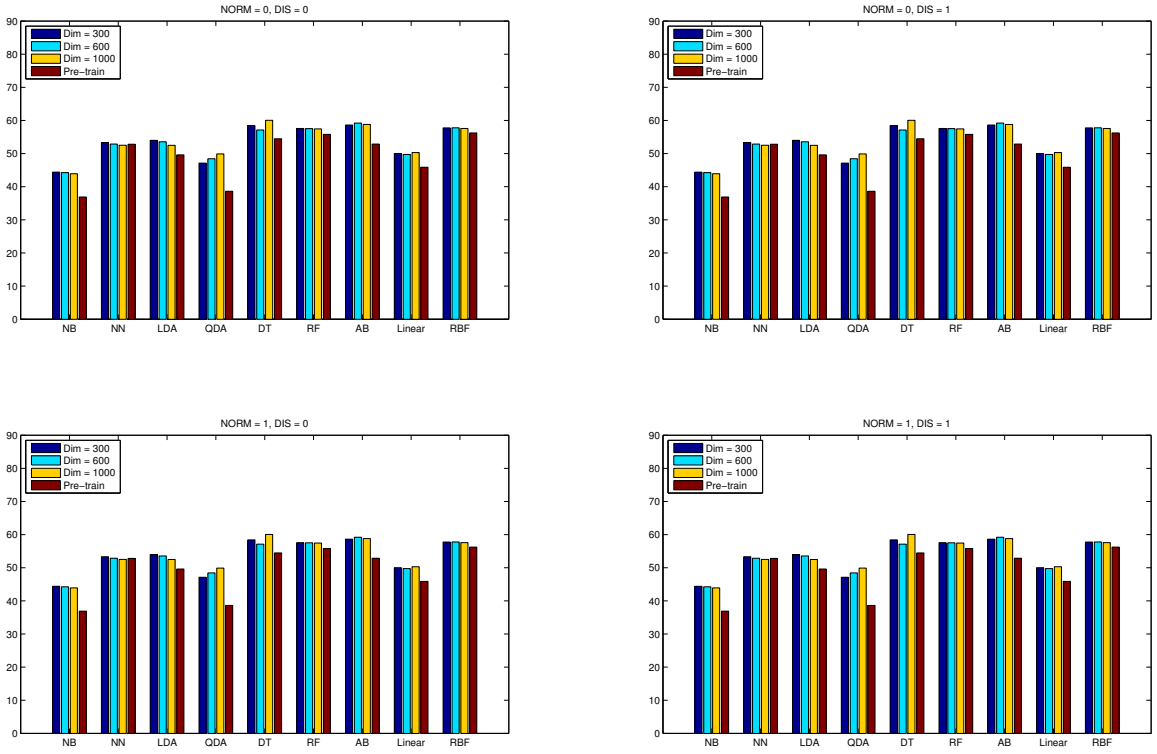

FIGURE 10: Accuracy of classifiers on word2vec model

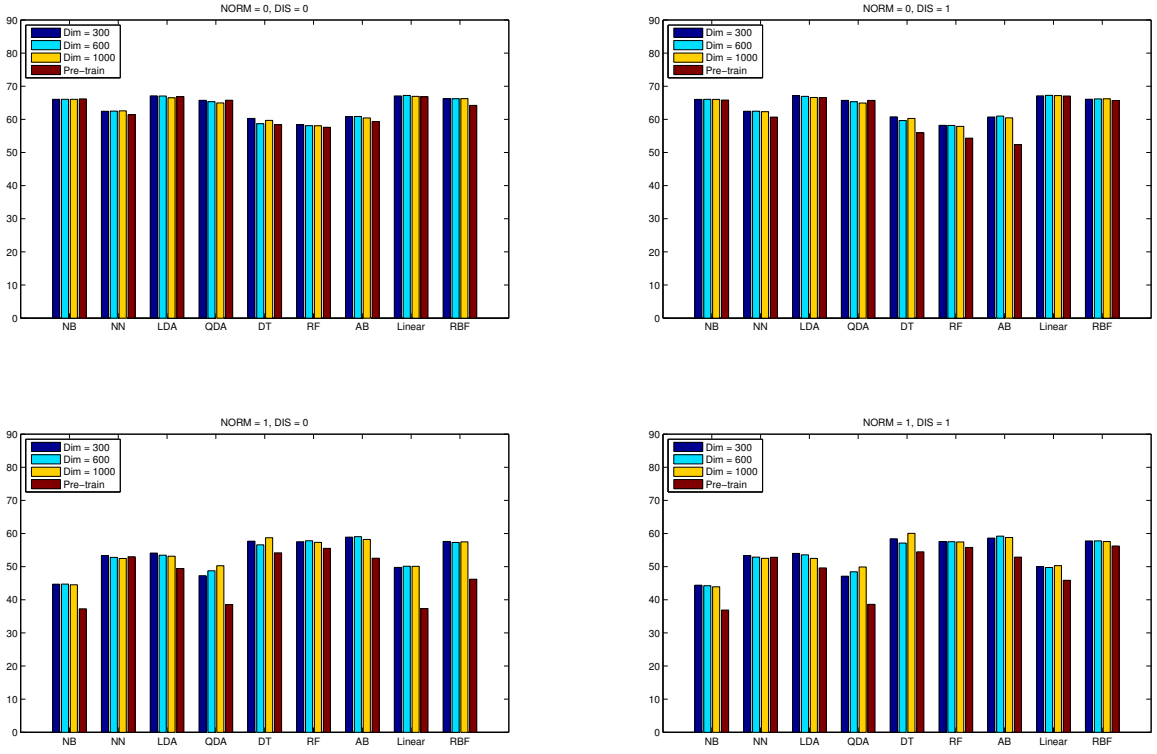

FIGURE 11: F1 of classifiers on word2vec model

5.2 Experiments of doc2vec

The problem of word2vec is that although the vectors can contain some position information during training word2vec model, the word order has been lost during summing up, resulting in different tweets can share the exactly same vector as long as the words they used are same. And this may also cause two tweets that contain very different words but the vectors of them are still same or in closer distance. This problem can be solved by using doc2vec.

In 2014, Q. Le et al. [24] proposed doc2vec that can learn continuous distributed vector representations for variable-length texts, ranging from sentences to documents.

Doc2vec approach is inspired by word2vec. In word2vec model, during the training of word vectors, this model will try to maximize the accuracy of prediction of next word in sentence by the words before this word. This idea has been taken into doc2vec. The difference between this two models is that in doc2vec, every document has also been mapped to a unique vector and will be trained together during maximization.

We used doc2vec as another way to generate vectors for users. In our experiments, we used Gensim [31], which contains an implementation of Paragraph Vector based on [24]. For each user in our sampled dataset, the tweets he sent have been merged into one document and then we trained document vector based on the merged tweets. We also trained three different dimensions (300, 600, 1,000) to test the differences of the results between Document Vector models. The parameters of training are same as the parameters using in word2vec.

Fig 12 and 13 show the accuracies and F1s of different classifiers on different dimensions of doc2vec models. We can figure out that SVM with RBF kernel on 1000 dimension of doc2vec model achieves best accuracy (73.28%) and best F1 (73.39%). This result is better than all the results of classification on word2vec model.

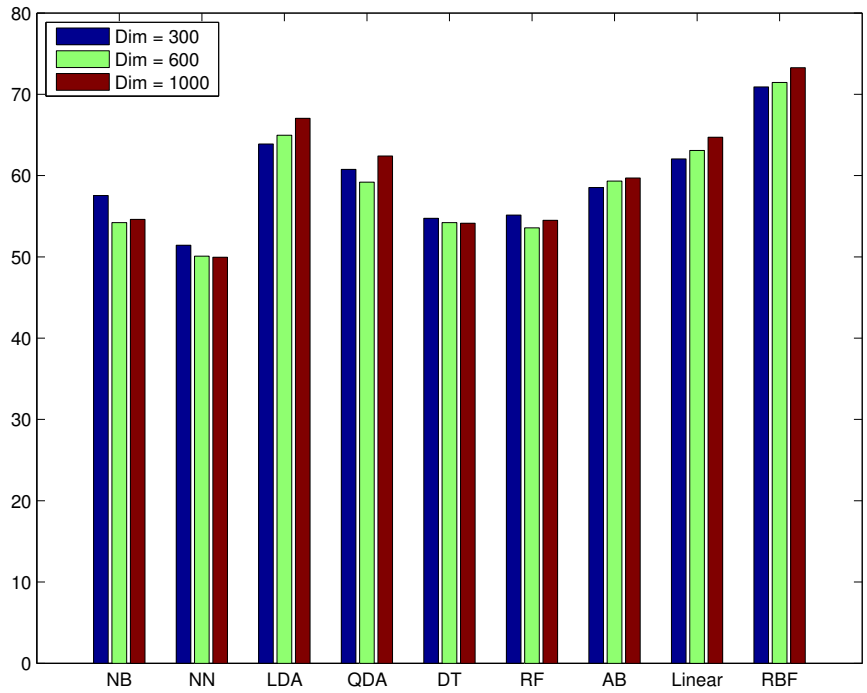

FIGURE 12: Accuracy of classifiers on doc2vec model

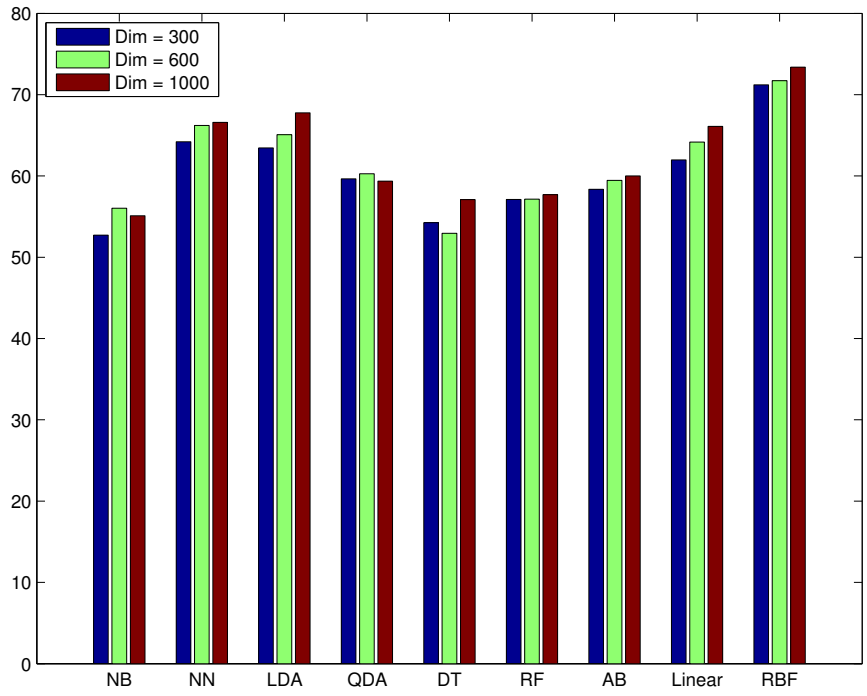

FIGURE 13: F1 of classifiers on doc2vec model

We also tried to visualize the users and tried to figure out the clusters and relation-

ships among users based on the 1000 dimension Paragraph Vector result. We reduced the dimension into 2D by t-Distributed Stochastic Neighbor Embedding (t-SNE) [32]. t-SNE is a faster dimension reduction algorithm that accepts a high-dimension matrix and then outputs a 2D matrix. When converting, t-SNE will first compute the distances between vectors and then it will build a similarity tree based on the distance matrix. It will then train a set of 2D vectors which can also satisfy the similarity tree so that these 2D vectors can still keep the structure of the high dimension vectors.

During the conversion, the order of rows won't be changed so that the user that the row represents is still the same. In our experiment, the t-SNE we used is an open source implementation (bhtsne) [33] that is based on variants of the Barnes-Hut algorithm and the dual-tree algorithm, whose time complexity is $O(NlogN)$. The input parameters of bhtsne are that the perplexity is 30.0 and θ is 0.5.

Fig. 14 shows the 2D matrix result of t-SNE. We can directly see that there are some small clusters of suspended users located around the borders. And in the center, the left part consists mainly of suspended users while the right part consists mainly of non-suspended users.

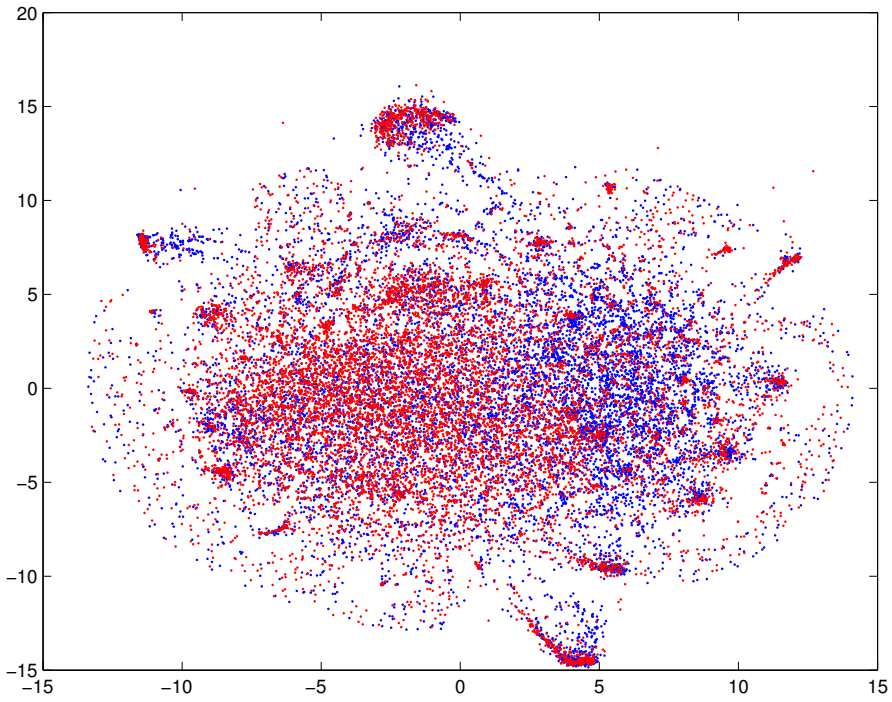

FIGURE 14: Visualizations of Users

CHAPTER 6

Classification using bad words

In previous section, we found that there are a lot of bad words in top features selected by PMI and WAPMI. This is a strong evidence that many users were suspended because of their abusive activities. So we explored further in the bad words they used.

First we collected a list of bad words from the term blocking lists of Google [34] and Facebook [35]. After merging these two lists, we got 487 bad words which can be found in our dataset. Table 26 contains top 10 bad words sorted by token frequency in suspended user dataset.

No.	Word	STF	NTF	SUF	NUF
1	f***	44276	16684	2876	1476
2	s***	35717	14140	2970	1622
3	d***	29031	14510	3077	1780
4	a**	22691	9251	2845	1526
5	s***	20035	9939	3619	2004
6	h***	19190	9773	3235	1853
7	b****	17779	6362	2465	1255
8	k***	15126	7909	3189	1803
9	w**	14441	6810	2479	1333
10	s*****	14234	7159	3068	1620

TABLE 26: Top 10 Bad Words Sorted by Frequency in Suspended User Dataset, STF = Suspended Token Frequency, NTF = Non-Suspended Token Frequency, SUF = Suspended User Frequency, NUF = Non-Suspended User Frequency

However, there is a challenge when analyzing the using of bad words when sending tweets, which is we cannot know the exactly reason of suspending of user. So even every tweet of user includes bad words, we still cannot say this user is suspended

because of abusing. But when looking at the whole picture, the number of suspending users who sending bad words should be larger than that number of non-suspended users. Fig. 15 shows the counts of users who sent top bad words. We can clearly figure out that the number of suspend users using top bad words is as twice as larger than that number of non-suspended users.

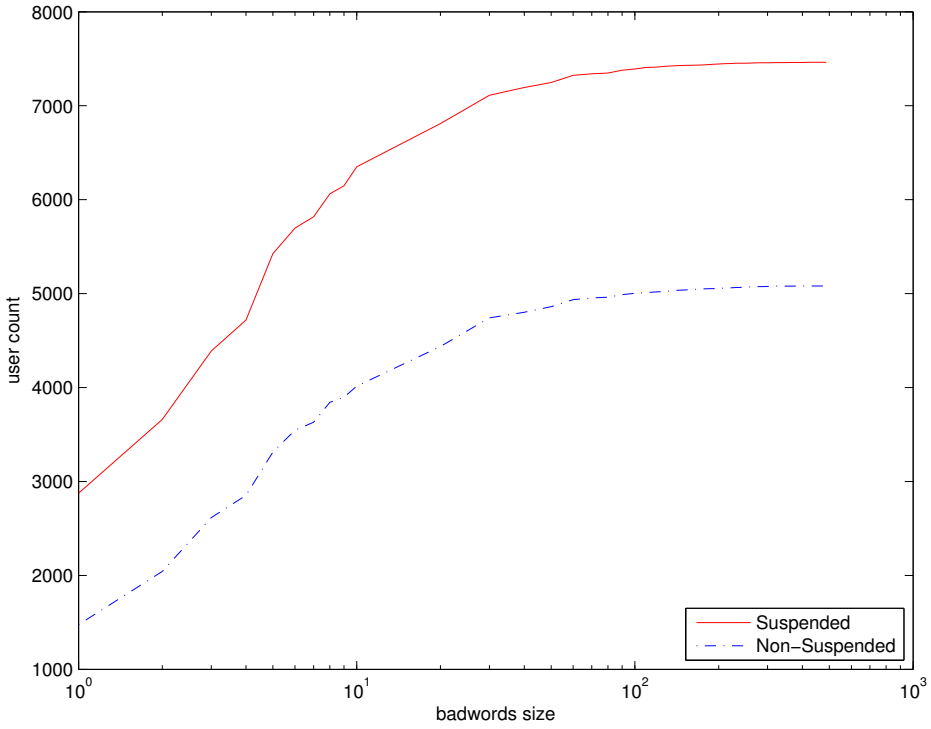

FIGURE 15: The number of users who sent top bad words

This character can also be detected by seeing Fig 16. In this figure, we plotted all the unigrams located by frequency in suspended user dataset and non-suspended user dataset. This is because the total frequency of tokens in both datasets are different, the directly comparison of raw frequencies is meaningless. The frequency has been normalized by,

$$Norm(N_{t,c}) = \frac{N_{t,c}}{N_c}$$

where $N_{t,c}$ is the count of gram t occurring in class c and N_c is the total gram frequency in class c. The red circles are bad words. The features below green line represent suspended users tend to use it more often and the features above green line represent normal users tend to use it more often. It shows that bad words are more likely to be used by suspended users, especially top bad words.

We trained a 300 dimension word2vec using the tokens of which frequency is larger than 5 to get deeper inside the relations between bad words and other tokens. Fig. 17 is plotted by t-sne which can reduce the 300 dimension word vectors into 2 dimension so that we can visualize all the words. The red markers are bad words and there is one big cluster of bad words located in the left center and a small cluster of bad words located in the right center. Because the vectors trained by word2vec can carry the position and meaning information, it should be highly possible that the other words which are close enough to the bad words can also be considered as bad words. So we computed the score between feature t and bad word w, which is defined as the token frequency of bad word in suspended user dataset multiplies the cosine similarity of feature word vector and bad word vector.

$$BadScore(t, w) = N_{w,s} \frac{V_t \cdot V_w}{|V_t||V_w|}$$

Table 27 listed top 10 most similar hidden bad words. These hidden bad words are not on the common bad words list.

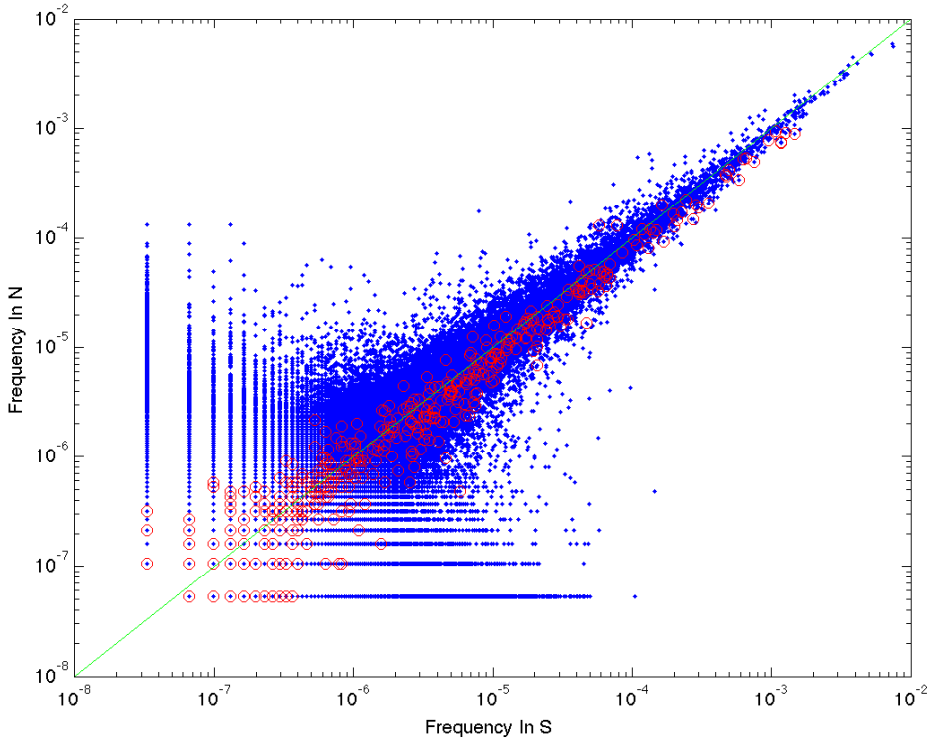

FIGURE 16: Bad words located by normalized token frequency

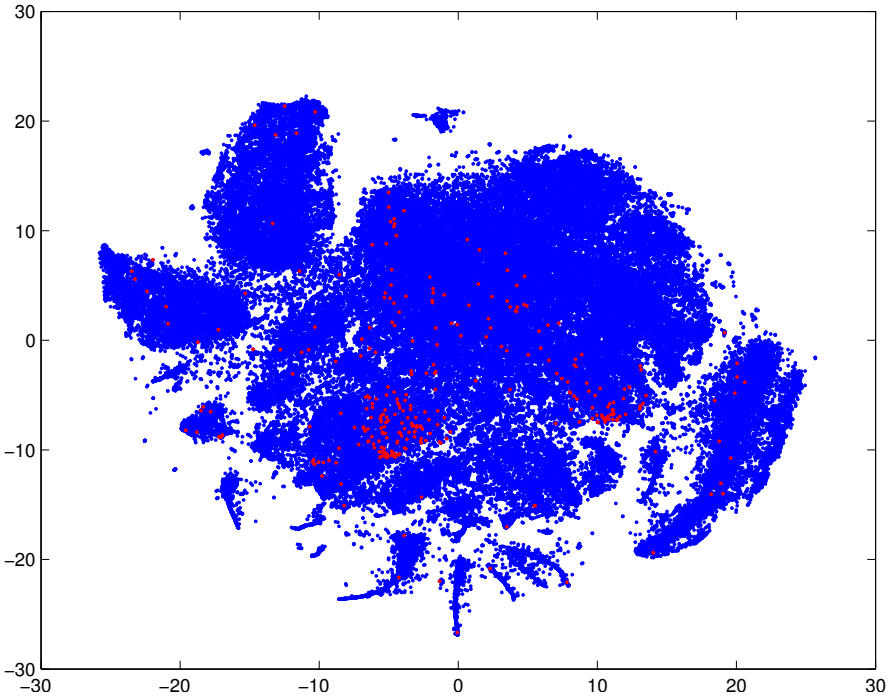

FIGURE 17: Bad word vectors trained by word2vec

No.	Word	COS	Score	Similar To	STF	NTF	SUF	NUF
1	f*****	0.64	28286.24	f***	334	362	70	37
2	c****	0.64	28170.08	f***	213	111	123	76
3	e**	0.63	28039.53	f***	1189	522	453	202
4	a*****	0.61	26929.79	f***	1951	805	876	386
5	s*	0.73	26090.08	s***	1264	746	595	337
6	e*	0.59	25921.04	f***	1432	663	519	248
7	s***	0.71	25439.94	s***	814	350	144	74
8	s****	0.57	25316.79	f***	4975	2206	1613	825
9	e****	0.57	25118.44	f***	1055	422	391	166
10	f******	0.55	24366.71	f***	976	436	405	198

TABLE 27: Top 10 Hidden Bad Words, COS = Cosine Similarity between hidden word and bad word, STF = Suspended Token Frequency, NTF = Non-Suspended Token Frequency, SUF = Suspended User Frequency, NUF = Non-Suspended User Frequency

In order to detect whether bad words are the reasons why the users were sus-

pended, we tried to classify them based on the bad words frequency of these users. Because this time we only got little number of features, so we can try a lot of classifiers. We used scikit-learn library [36], which contains many different classifiers such as Naive Bayes, Nearest Neighbours and Random Forest. All the parameters of the classifiers remained default. The classifiers we used in our experiments listed in Table 29. Table 28 shows the result of classification. We can figure out when using decision tree or random forest, we can achieve about 60% accuracy and 74% F1, which is better than using MNB with full features. This is also an evidence to prove that bad words are one of main reasons of suspending.

CLR	TP	FP	FN	TN	Precision	Recall	Accuracy	F1
NB	820	6643	371	4710	68.85 %	10.99 %	44.08 %	18.95 %
NN	6191	1272	4207	874	59.54 %	82.96 %	56.32 %	69.32 %
LDA	7067	396	4808	273	59.51 %	94.69 %	58.51 %	73.09 %
QDA	847	6616	410	4671	67.38 %	11.35 %	43.99 %	19.43 %
DT	7279	184	4947	134	59.54 %	97.53 %	59.10 %	73.94 %
RF	7462	1	5081	0	59.49 %	99.99 %	59.49 %	74.60 %
AB	7086	377	4761	320	59.81 %	94.95 %	59.04 %	73.39 %

TABLE 28: Classification Results using Bad Words

CHAPTER 7

Conclusion

In this thesis, we analysed the suspended users in Twitter and studied several approaches to predict whether a user will be suspended or not. First, we took a review of the some related works. Benevenuto et al. [4] and Moh et al. [5] gave a good work on classification of spammers with high accuracy. However the works from them meet a problem that the size of dataset is too small. They only focus on a narrow subdomain of twitter, which means their work cannot directly be scaled to the whole twitter social network. Moreover, the feature set they selected can be easily manipulated by spammers to act like a real user so that their approach cannot detect them.

In our research, we collected a large dataset of suspended users and analysed this dataset to reveal the different between suspended users and normal users. We tried to classify them by Naive Bayes classifier, together with several feature selection methods on n-gram models derived from the tweets, resulting in 76.75% accuracy and 78.54% F1. We also tried different word embedding methods to convert users into vectors. We tried some classifiers on converted user vectors. When using SVM with RBF kernel function, we achieved 73.28% accuracy and 73.39% F1 on 1000 dimension user vector trained by Paragraph Vector method.

After analyzing the characteristic of bad words using in suspended users, we found the number of bad words using users is as twice larger as that number of normal users. And we also introduced Badscore to evaluate probability of whether a word can be a bad word.

REFERENCES

[1] The twitter usage report. `https://about.twitter.com/company`.

[2] Zi Chu, Indra Widjaja, and Haining Wang. Detecting social spam campaigns on twitter. In *Applied Cryptography and Network Security*, pages 455–472. Springer, 2012.

[3] Michael Mccord and M Chuah. Spam detection on twitter using traditional classifiers. In *Autonomic and trusted computing*, pages 175–186. Springer, 2011.

[4] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, page 12, 2010.

[5] Teng-Sheng Moh and Alexander J Murmann. Can you judge a man by his friends?-enhancing spammer detection on the twitter microblogging platform using friends and followers. In *Information Systems, Technology and Management*, pages 210–220. Springer, 2010.

[6] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 35–47. ACM, 2010.

[7] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 447–462. IEEE, 2011.

[8] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 1–9. ACM, 2010.

[9] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers: social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442. ACM, 2010.

[10] Kurt Thomas, Chris Grier, Dawn Song, and Vern Paxson. Suspended accounts in retrospect: an analysis of twitter spam. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 243–258. ACM, 2011.

[11] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st international conference on World Wide Web*, pages 61–70. ACM, 2012.

[12] Beate Krause, Christoph Schmitz, Andreas Hotho, and Gerd Stumme. The anti-social tagger: detecting spam in social bookmarking systems. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pages 61–68. ACM, 2008.

[13] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and P Krishna Gummadi. Measuring user influence in twitter: The million follower fallacy. *ICWSM*, 10(10-17):30, 2010.

[14] Twitter dataset homepage. `http://twitter.mpi-sws.org`.

[15] Tianyin Xu, Yang Chen, Lei Jiao, Ben Y Zhao, Pan Hui, and Xiaoming Fu. Scaling microblogging services with divergent traffic demands. In *Proceedings of the 12th International Middleware Conference*, pages 20–39. International Federation for Information Processing, 2011.

[16] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[17] Stop words homepage. `http://xpo6.com/list-of-english-stop-words/`.

[18] Martin F Porter. Snowball: A language for stemming algorithms, 2001.

[19] Porter2 stemmer homepage. `https://github.com/smassung/porter2_stemmer`.

[20] Karl-Rudolf Koch. *Bayes Theorem*. Springer, 1990.

[21] Karl-Michael Schneider. Weighted average pointwise mutual information for feature selection in text categorization. In *Knowledge Discovery in Databases: PKDD 2005*, pages 252–263. Springer, 2005.

[22] Kurt Thomas and David M Nicol. The koobface botnet and the rise of social malware. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 63–70. IEEE, 2010.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[24] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[25] Bai Xue, Chen Fu, and Zhan Shaobin. A new clustering model based on word2vec mining on sina weibo users tags. 2014.

[26] Ayah Zirikly and Mona Diab. Named entity recognition for arabic social media. In *Proceedings of naacl-hlt*, pages 176–185, 2015.

[27] Scharolta Katharina Siencnik. Adapting word2vec to named entity recognition. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 239–243, 2015.

[28] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[29] Zhihua Zhang, Guoshun Wu, and Man Lan. Ecnu: Multi-level sentiment analysis on twitter using traditional linguistic features and word embedding features. *SemEval-2015*, page 561, 2015.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[31] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[32] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[33] bhtsne homepage. `https://github.com/lvdmaaten/bhtsne/`.

[34] Google bad words list. `https://gist.github.com/jamiew/1112488`.

[35] Facebook bad words list. `http://www.frontgatemedia.com/a-list-of-723-bad-words-to-blacklist-and-how-to-use-facebooks-moderation-tool/`.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

APPENDIX A

Classifiers used in experiments

Name	Short Name	Description
Gaussian Naive Bayes	NB	Gaussian Naive Bayes Classifier. The likelihood of features is assumed to be Gaussian distribution.
Nearest Neighbour	NN	k-nearest neighbour classifier with k = 10. Classify test case by vote of k-nearest neighbour of it.
Linear Discriminant Analysis	LDA	A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.
Quadratic Discriminant Analysis	QDA	A classifier with a quadratic decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.
Decision Tree	DT	A classifier that trains a model which can predict the value of a target variable by learning simple decision rules inferred from the data features.
Random Forest	RF	A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset.
Adaptive Boosting	AB	It is also a meta-estimator that begins by fitting a classifier (decision tree) on the original dataset and then fits additional copies of the classifier on the same dataset
SVM with Linear Kernel	Linear	Support vector machines with linear function kernel
SVM with RBF Kernel	RBF	Support vector machines with radial basis function kernel

TABLE 29: Classifiers in Experiments

APPENDIX B

Classification results of word2vec and doc2vec

CLR	N	D	TP	FP	FN	TN	Precision	Recall	Accuracy	F1
NB	0	0	10348	986	9654	1680	51.73 %	91.30 %	53.06 %	66.05 %
NN	0	0	7547	3787	5291	6043	58.79 %	66.59 %	59.95 %	62.44 %
LDA	0	0	9796	1538	8073	3261	54.82 %	86.43 %	57.60 %	67.09 %
QDA	0	0	10034	1300	9155	2179	52.29 %	88.53 %	53.88 %	65.75 %
DT	0	0	6548	4786	3845	7489	63.00 %	57.77 %	61.92 %	60.28 %
RF	0	0	6012	5322	3235	8099	65.02 %	53.04 %	62.25 %	58.42 %
AB	0	0	6524	4810	3595	7739	64.47 %	57.56 %	62.92 %	60.82 %
Linear	0	0	10389	945	9260	2074	52.87 %	91.66 %	54.98 %	67.06 %
RBF	0	0	9319	2015	7480	3854	55.47 %	82.22 %	58.11 %	66.25 %
NB	1	0	3784	7550	1813	9521	67.61 %	33.39 %	58.70 %	44.70 %
NN	1	0	5455	5879	3672	7662	59.77 %	48.13 %	57.87 %	53.32 %
LDA	1	0	5395	5939	3221	8113	62.62 %	47.60 %	59.59 %	54.09 %
QDA	1	0	4042	7292	1726	9608	70.08 %	35.66 %	60.22 %	47.27 %
DT	1	0	6196	5138	3952	7382	61.06 %	54.67 %	59.90 %	57.69 %
RF	1	0	6010	5324	3552	7782	62.85 %	53.03 %	60.84 %	57.52 %
AB	1	0	6242	5092	3617	7717	63.31 %	55.07 %	61.58 %	58.91 %
Linear	1	0	4558	6776	2428	8906	65.24 %	40.22 %	59.40 %	49.76 %
RBF	1	0	5726	5608	2817	8517	67.03 %	50.52 %	62.83 %	57.61 %
NB	0	1	10348	986	9661	1673	51.72 %	91.30 %	53.03 %	66.03 %
NN	0	1	7558	3776	5315	6019	58.71 %	66.68 %	59.90 %	62.44 %
LDA	0	1	9817	1517	8100	3234	54.79 %	86.62 %	57.57 %	67.12 %
QDA	0	1	10031	1303	9160	2174	52.27 %	88.50 %	53.84 %	65.72 %
DT	0	1	6692	4642	4007	7327	62.55 %	59.04 %	61.84 %	60.75 %
RF	0	1	5972	5362	3220	8114	64.97 %	52.69 %	62.14 %	58.19 %
AB	0	1	6522	4812	3628	7706	64.26 %	57.54 %	62.77 %	60.71 %
Linear	0	1	10383	951	9236	2098	52.92 %	91.61 %	55.06 %	67.09 %
RBF	0	1	9311	2023	7534	3800	55.27 %	82.15 %	57.84 %	66.08 %
NB	1	1	3748	7586	1800	9534	67.56 %	33.07 %	58.59 %	44.40 %
NN	1	1	5457	5877	3673	7661	59.77 %	48.15 %	57.87 %	53.33 %
LDA	1	1	5389	5945	3244	8090	62.42 %	47.55 %	59.46 %	53.98 %
QDA	1	1	4020	7314	1708	9626	70.18 %	35.47 %	60.20 %	47.12 %
DT	1	1	6437	4897	4261	7073	60.17 %	56.79 %	59.60 %	58.43 %
RF	1	1	6005	5329	3521	7813	63.04 %	52.98 %	60.96 %	57.57 %
AB	1	1	6226	5108	3683	7651	62.83 %	54.93 %	61.22 %	58.62 %
Linear	1	1	4620	6714	2515	8819	64.75 %	40.76 %	59.29 %	50.03 %
RBF	1	1	5743	5591	2808	8526	67.16 %	50.67 %	62.95 %	57.76 %

TABLE 30: Classification Result on Tweets Trained Word Vector (Dimension = 300)

CLR	N	D	TP	FP	FN	TN	Precision	Recall	Accuracy	F1
NB	0	0	10350	984	9654	1680	51.74 %	91.32 %	53.07 %	66.05 %
NN	0	0	7550	3784	5276	6058	58.86 %	66.61 %	60.03 %	62.50 %
LDA	0	0	9726	1608	7954	3380	55.01 %	85.81 %	57.82 %	67.04 %
QDA	0	0	9707	1627	8667	2667	52.83 %	85.64 %	54.59 %	65.35 %
DT	0	0	6225	5109	3646	7688	63.06 %	54.92 %	61.38 %	58.71 %
RF	0	0	5932	5402	3155	8179	65.28 %	52.34 %	62.25 %	58.10 %
AB	0	0	6563	4771	3669	7665	64.14 %	57.91 %	62.77 %	60.86 %
Linear	0	0	10269	1065	8955	2379	53.42 %	90.60 %	55.80 %	67.21 %
RBF	0	0	9330	2004	7509	3825	55.41 %	82.32 %	58.03 %	66.23 %
NB	1	0	3788	7546	1817	9517	67.58 %	33.42 %	58.70 %	44.73 %
NN	1	0	5348	5986	3578	7756	59.91 %	47.19 %	57.81 %	52.79 %
LDA	1	0	5316	6018	3241	8093	62.12 %	46.90 %	59.15 %	53.45 %
QDA	1	0	4264	7070	1894	9440	69.24 %	37.62 %	60.46 %	48.75 %
DT	1	0	5953	5381	3751	7583	61.35 %	52.52 %	59.71 %	56.59 %
RF	1	0	6068	5266	3586	7748	62.85 %	53.54 %	60.95 %	57.82 %
AB	1	0	6254	5080	3590	7744	63.53 %	55.18 %	61.75 %	59.06 %
Linear	1	0	4649	6685	2568	8766	64.42 %	41.02 %	59.18 %	50.12 %
RBF	1	0	5663	5671	2769	8565	67.16 %	49.96 %	62.77 %	57.30 %
NB	0	1	10349	985	9659	1675	51.72 %	91.31 %	53.04 %	66.04 %
NN	0	1	7553	3781	5293	6041	58.80 %	66.64 %	59.97 %	62.47 %
LDA	0	1	9733	1601	8015	3319	54.84 %	85.87 %	57.58 %	66.93 %
QDA	0	1	9703	1631	8676	2658	52.79 %	85.61 %	54.53 %	65.31 %
DT	0	1	6418	4916	3771	7563	62.99 %	56.63 %	61.68 %	59.64 %
RF	0	1	5946	5388	3167	8167	65.25 %	52.46 %	62.26 %	58.16 %
AB	0	1	6576	4758	3650	7684	64.31 %	58.02 %	62.91 %	61.00 %
Linear	0	1	10276	1058	8954	2380	53.44 %	90.67 %	55.83 %	67.24 %
RBF	0	1	9322	2012	7520	3814	55.35 %	82.25 %	57.95 %	66.17 %
NB	1	1	3731	7603	1798	9536	67.48 %	32.92 %	58.53 %	44.25 %
NN	1	1	5347	5987	3543	7791	60.15 %	47.18 %	57.96 %	52.88 %
LDA	1	1	5321	6013	3214	8120	62.34 %	46.95 %	59.30 %	53.56 %
QDA	1	1	4221	7113	1869	9465	69.31 %	37.24 %	60.38 %	48.45 %
DT	1	1	6072	5262	3854	7480	61.17 %	53.57 %	59.78 %	57.12 %
RF	1	1	6015	5319	3563	7771	62.80 %	53.07 %	60.82 %	57.53 %
AB	1	1	6301	5033	3648	7686	63.33 %	55.59 %	61.70 %	59.21 %
Linear	1	1	4611	6723	2597	8737	63.97 %	40.68 %	58.88 %	49.74 %
RBF	1	1	5718	5616	2739	8595	67.61 %	50.45 %	63.14 %	57.78 %

TABLE 31: Classification Result on Tweets Trained Word Vector (Dimension = 600)

CLR	N	D	TP	FP	FN	TN	Precision	Recall	Accuracy	F1
NB	0	0	10351	983	9655	1679	51.74 %	91.33 %	53.07 %	66.06 %
NN	0	0	7556	3778	5266	6068	58.93 %	66.67 %	60.10 %	62.56 %
LDA	0	0	9579	1755	7890	3444	54.83 %	84.52 %	57.45 %	66.51 %
QDA	0	0	9318	2016	8038	3296	53.69 %	82.21 %	55.65 %	64.96 %
DT	0	0	6464	4870	3859	7475	62.62 %	57.03 %	61.49 %	59.69 %
RF	0	0	5954	5380	3217	8117	64.92 %	52.53 %	62.07 %	58.07 %
AB	0	0	6514	4820	3706	7628	63.74 %	57.47 %	62.39 %	60.44 %
Linear	0	0	10101	1233	8750	2584	53.58 %	89.12 %	55.96 %	66.93 %
RBF	0	0	9320	2014	7477	3857	55.49 %	82.23 %	58.13 %	66.26 %
NB	1	0	3763	7571	1802	9532	67.62 %	33.20 %	58.65 %	44.54 %
NN	1	0	5294	6040	3554	7780	59.83 %	46.71 %	57.68 %	52.46 %
LDA	1	0	5263	6071	3205	8129	62.15 %	46.44 %	59.08 %	53.16 %
QDA	1	0	4535	6799	2165	9169	67.69 %	40.01 %	60.46 %	50.29 %
DT	1	0	6519	4815	4350	6984	59.98 %	57.52 %	59.57 %	58.72 %
RF	1	0	5971	5363	3530	7804	62.85 %	52.68 %	60.77 %	57.32 %
AB	1	0	6145	5189	3634	7700	62.84 %	54.22 %	61.08 %	58.21 %
Linear	1	0	4664	6670	2623	8711	64.00 %	41.15 %	59.00 %	50.09 %
RBF	1	0	5672	5662	2726	8608	67.54 %	50.04 %	63.00 %	57.49 %
NB	0	1	10347	987	9659	1675	51.72 %	91.29 %	53.04 %	66.03 %
NN	0	1	7525	3809	5297	6037	58.69 %	66.39 %	59.83 %	62.30 %
LDA	0	1	9611	1723	7925	3409	54.81 %	84.80 %	57.44 %	66.58 %
QDA	0	1	9317	2017	8048	3286	53.65 %	82.20 %	55.60 %	64.93 %
DT	0	1	6559	4775	3874	7460	62.87 %	57.87 %	61.84 %	60.27 %
RF	0	1	5909	5425	3175	8159	65.05 %	52.14 %	62.06 %	57.88 %
AB	0	1	6523	4811	3724	7610	63.66 %	57.55 %	62.35 %	60.45 %
Linear	0	1	10138	1196	8709	2625	53.79 %	89.45 %	56.30 %	67.18 %
RBF	0	1	9338	1996	7536	3798	55.34 %	82.39 %	57.95 %	66.21 %
NB	1	1	3690	7644	1779	9555	67.47 %	32.56 %	58.43 %	43.92 %
NN	1	1	5294	6040	3535	7799	59.96 %	46.71 %	57.76 %	52.51 %
LDA	1	1	5179	6155	3217	8117	61.68 %	45.69 %	58.66 %	52.50 %
QDA	1	1	4481	6853	2146	9188	67.62 %	39.54 %	60.30 %	49.90 %
DT	1	1	6860	4474	4654	6680	59.58 %	60.53 %	59.73 %	60.05 %
RF	1	1	5989	5345	3529	7805	62.92 %	52.84 %	60.85 %	57.44 %
AB	1	1	6228	5106	3622	7712	63.23 %	54.95 %	61.50 %	58.80 %
Linear	1	1	4717	6617	2701	8633	63.59 %	41.62 %	58.89 %	50.31 %
RBF	1	1	5688	5646	2736	8598	67.52 %	50.19 %	63.02 %	57.58 %

TABLE 32: Classification Result on Tweets Trained Word Vector (Dimension = 1,000)

CLR	N	D	TP	FP	FN	TN	Precision	Recall	Accuracy	F1
NB	0	0	10359	975	9619	1715	51.85 %	91.40 %	53.26 %	66.17 %
NN	0	0	7330	4004	5202	6132	58.49 %	64.67 %	59.39 %	61.43 %
LDA	0	0	9720	1614	8021	3313	54.79 %	85.76 %	57.50 %	66.86 %
QDA	0	0	9967	1367	9010	2324	52.52 %	87.94 %	54.22 %	65.76 %
DT	0	0	6150	5184	3577	7757	63.23 %	54.26 %	61.35 %	58.40 %
RF	0	0	5853	5481	3131	8203	65.15 %	51.64 %	62.01 %	57.61 %
AB	0	0	6249	5085	3488	7846	64.18 %	55.13 %	62.18 %	59.31 %
Linear	0	0	10566	768	9724	1610	52.07 %	93.22 %	53.71 %	66.82 %
RBF	0	0	8719	2615	7099	4235	55.12 %	76.93 %	57.15 %	64.22 %
NB	1	0	2923	8411	1421	9913	67.29 %	25.79 %	56.63 %	37.29 %
NN	1	0	5519	5815	3971	7363	58.16 %	48.69 %	56.83 %	53.01 %
LDA	1	0	4796	6538	3276	8058	59.42 %	42.32 %	56.71 %	49.43 %
QDA	1	0	3012	8322	1278	10056	70.21 %	26.57 %	57.65 %	38.56 %
DT	1	0	5491	5843	3450	7884	61.41 %	48.45 %	59.00 %	54.17 %
RF	1	0	5636	5698	3331	8003	62.85 %	49.73 %	60.17 %	55.52 %
AB	1	0	5143	6191	3100	8234	62.39 %	45.38 %	59.01 %	52.54 %
Linear	1	0	2995	8339	1705	9629	63.72 %	26.42 %	55.69 %	37.36 %
RBF	1	0	4021	7313	2046	9288	66.28 %	35.48 %	58.71 %	46.22 %
NB	0	1	10331	1003	9723	1611	51.52 %	91.15 %	52.68 %	65.83 %
NN	0	1	7269	4065	5358	5976	57.57 %	64.13 %	58.43 %	60.67 %
LDA	0	1	9908	1426	8525	2809	53.75 %	87.42 %	56.10 %	66.57 %
QDA	0	1	10051	1283	9209	2125	52.19 %	88.68 %	53.71 %	65.71 %
DT	0	1	5626	5708	3156	8178	64.06 %	49.64 %	60.90 %	55.94 %
RF	0	1	5324	6010	2945	8389	64.39 %	46.97 %	60.49 %	54.32 %
AB	0	1	4933	6401	2562	8772	65.82 %	43.52 %	60.46 %	52.40 %
Linear	0	1	10418	916	9322	2012	52.78 %	91.92 %	54.84 %	67.05 %
RBF	0	1	9008	2326	7081	4253	55.99 %	79.48 %	58.50 %	65.70 %
NB	1	1	2883	8451	1411	9923	67.14 %	25.44 %	56.49 %	36.90 %
NN	1	1	5481	5853	3941	7393	58.17 %	48.36 %	56.79 %	52.81 %
LDA	1	1	4814	6520	3256	8078	59.65 %	42.47 %	56.87 %	49.62 %
QDA	1	1	3007	8327	1223	10111	71.09 %	26.53 %	57.87 %	38.64 %
DT	1	1	5530	5804	3437	7897	61.67 %	48.79 %	59.23 %	54.48 %
RF	1	1	5671	5663	3321	8013	63.07 %	50.04 %	60.37 %	55.80 %
AB	1	1	5193	6141	3118	8216	62.48 %	45.82 %	59.15 %	52.87 %
Linear	1	1	4020	7314	2175	9159	64.89 %	35.47 %	58.14 %	45.87 %
RBF	1	1	5465	5869	2637	8697	67.45 %	48.22 %	62.48 %	56.24 %

TABLE 33: Classification Result on Pretrained Word Vector (Dimension = 300)

CLR	Dim	TP	FP	FN	TN	Precision	Recall	Accuracy	F1
NB	300	5364	5970	3654	7680	59.48 %	47.33 %	57.54 %	52.71 %
NN	300	9877	1457	9553	1781	50.83 %	87.14 %	51.43 %	64.21 %
LDA	300	7106	4228	3958	7376	64.23 %	62.70 %	63.89 %	63.45 %
QDA	300	6572	4762	4131	7203	61.40 %	57.98 %	60.77 %	59.65 %
DT	300	6083	5251	5008	6326	54.85 %	53.67 %	54.74 %	54.25 %
RF	300	6766	4568	5599	5735	54.72 %	59.70 %	55.15 %	57.10 %
AB	300	6585	4749	4652	6682	58.60 %	58.10 %	58.53 %	58.35 %
Linear	300	7014	4320	4284	7050	62.08 %	61.88 %	62.04 %	61.98 %
RBF	300	8159	3175	3421	7913	70.46 %	71.99 %	70.90 %	71.21 %
NB	600	6613	4721	5658	5676	53.89 %	58.35 %	54.21 %	56.03 %
NN	600	11089	245	11070	264	50.04 %	97.84 %	50.08 %	66.22 %
LDA	600	7399	3935	4009	7325	64.86 %	65.28 %	64.96 %	65.07 %
QDA	600	7016	4318	4931	6403	58.73 %	61.90 %	59.20 %	60.27 %
DT	600	5840	5494	4884	6450	54.46 %	51.53 %	54.22 %	52.95 %
RF	600	7018	4316	6209	5125	53.06 %	61.92 %	53.57 %	57.15 %
AB	600	6760	4574	4645	6689	59.27 %	59.64 %	59.33 %	59.46 %
Linear	600	7490	3844	4523	6811	62.35 %	66.08 %	63.09 %	64.16 %
RBF	600	8202	3132	3337	7997	71.08 %	72.37 %	71.46 %	71.72 %
NB	1000	6314	5020	5267	6067	54.52 %	55.71 %	54.62 %	55.11 %
NN	1000	11304	30	11313	21	49.98 %	99.74 %	49.96 %	66.59 %
LDA	1000	7848	3486	3983	7351	66.33 %	69.24 %	67.05 %	67.76 %
QDA	1000	6223	5111	3411	7923	64.59 %	54.91 %	62.41 %	59.36 %
DT	1000	6917	4417	5979	5355	53.64 %	61.03 %	54.14 %	57.09 %
RF	1000	7034	4300	6013	5321	53.91 %	62.06 %	54.50 %	57.70 %
AB	1000	6851	4483	4652	6682	59.56 %	60.45 %	59.70 %	60.00 %
Linear	1000	7795	3539	4460	6874	63.61 %	68.78 %	64.71 %	66.09 %
RBF	1000	8353	2981	3077	8257	73.08 %	73.70 %	**73.28** %	**73.39** %

TABLE 34: Classification Result on doc2vec

VITA AUCTORIS

NAME:	Xiutian Cui
PLACE OF BIRTH:	Harbin, Heilongjiang province, China
YEAR OF BIRTH:	1987
EDUCATION:	Beijing University of Technology, B.Eng., Computer Science and Technology, Beijing, China, 2011
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2016