Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal

Volume 4 | Issue 2 Article 5

July 2017

Identifying Twitter Spam by Utilizing Random Forests

Humza S. Haider
University of Minnesota, Morris

Identifying Twitter Spam by Utilizing Random Forests

Cover Page Footnote

Identifying Twitter Spam by Utilizing Random Forests

Humza S. Haider
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
haide043@morris.umn.edu

ABSTRACT

The use of Twitter has rapidly grown since the first tweet in 2006. The number of spammers on Twitter shows a similar increase. Classifying users into spammers and non-spammers has been heavily researched, and new methods for spam detection are developing rapidly. One of these classification techniques is known as random forests. We examine three studies that employ random forests using user based features, geo-tagged features, and time dependent features. Each study showed high accuracy rates and F-measures with the exception of one model that had a test set with a more realistic proportion of spam relative to typical testing procedures. These studies suggest that random forests, in combination with unique feature selection can be used to identify spam and spammers with high accuracy but may have shortcomings when applied to real world situations.

Keywords

random forests,decision trees, spam, machine learning

1. INTRODUCTION

Twitter has become one of the most widely used social media services across the world averaging roughly 500 million tweets per day [6]. As a result, the number of malicious users on Twitter has also expanded. On Twitter, *spam* is defined as any unsolicited, repeated actions that negatively impact other users. On Twitter, spam can manifest itself as links to harmful websites (such as phishing or malware websites) or distribution of pornographic materials.

The data used to identify spam tweets and spammers is made up of *features* also known as *variables*. Features are a way of categorizing a dataset into mutually exclusive groups such as tweets into spam and not spam or users into spammers and non-spammers. Another feature could be an indicator of whether or not a user has reported a tweet as spam. This feature, *Reported*, takes on the values *Reported* = yes and *Reported* = no. Note that a reported tweet doesn't imply the tweet is spam, just that it has been reported. All features in this paper are denoted with italics.

Spammers typically share traits within their tweets and their accounts. For example, in order to draw more attention to their tweets, a spammer is likely to have tweets containing many *mentions*, a way of bringing a user's attention to a tweet. Specifically, if we saw *Number of Mentions* = 5, it would imply the spammer mentioned 5 other users in their tweet. If a user has many mentions across their tweets the likelihood that they are a spammer increases. By combining multiple features, models can be constructed to identify spammers and spam tweets with a high degree of accuracy.

Machine learning is a specific type of artificial intelligence that allows computers to learn without being specifically programmed. Techniques in the field range from classification analysis to pattern recognition. Our paper explores the machine learning technique known as *random forests*. Introduced in 2001 by Leo Breiman, random forests have become one of the most widely used methods for classification analysis. We examine three papers which are similar in that each use a random forest to identify either spam or spammers on Twitter but are different in their feature selection.

Authors, Chen et al., [2] use features directly available from a tweet itself and features tied to a user's account. For example, they identify the the number of hashtags in a tweet and then also examine the number of followers that the user has. Guo and Chen [3] use similar features, but also include geographical information tied to a user and the user's tweets. For example, if a user travels large distances in relatively short periods between tweets they may be a spammer. Lastly, Washha et al. [6] make use of time-dependent features to identify spammers. One such feature is how tweeting diversity may change over time. If a user has little-to-no diversity in their tweets, they are likely tweeting a single message many times, making them a spammer.

First, a discussion of the concept and details of a random forest is given in Section 2. From there, the three studies' features are presented in Section 3. In Section 4, the analysis of each study will be presented and a comparison of the analyses will be made. Lastly, concluding remarks can be found in Section 5.

2. BACKGROUND

Before we begin to discuss the intricacies of a random forest, we will first consider a single tree. Random forests are constructed from a combination of *decision trees*. A decision tree is a method of classifying a feature of interest, denoted T, through the use of other features available. In the case of detecting spam tweets, knowing whether a URL is present in a tweet, the users account age, and if the tweet has been reported as spam could all be assessed before deciding on whether a tweet is spam or not spam. Table 1 gives examples of these features.

UMM CSci Senior Seminar Conference, April 2017 Morris, MN.

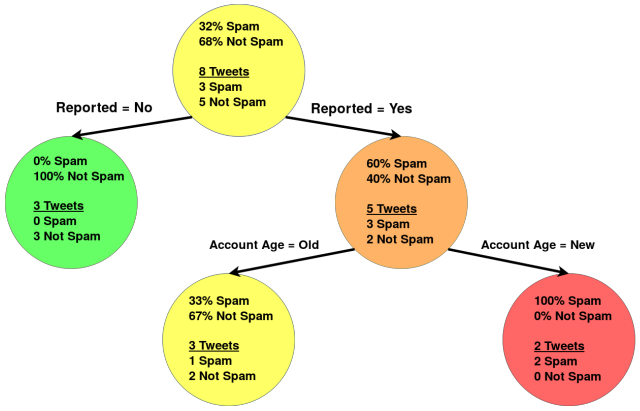
The data in Table 1 have been artificially constructed for the example and do not come from a real data set. Table 1 includes five features in total: URL, $AccountAge$, $Reported$, the feature of interest ($Class$), and an identification feature (ID). The ID feature lets us uniquely identify rows, where each row serves as a single observation (tweet). The feature URL identifies whether or not a URL is present in a tweet, $AccountAge$ specifies whether an account is old or new, $Reported$ specifies if the tweet has been reported as spam or not, and $Class$ determines whether or not the tweet is spam.

Table 1: Example data for classifying a tweet as spam or not spam.

ID	URL	Account Age	Reported	Class
e_1	No	Old	Yes	Not Spam
e_2	Yes	Old	Yes	Not Spam
e_3	No	New	Yes	Spam
e_4	No	Old	Yes	Spam
e_5	No	New	No	Not Spam
e_6	Yes	New	Yes	Spam
e_7	Yes	New	No	Not Spam
e_8	No	Old	No	Non Spam

Using the data in Table 1 a decision tree has been constructed, as seen in Figure 1. Each of the circles in the decision tree is referred to as a *node*. Within each node of the tree are the counts and proportions of Spam and Not Spam, *e.g.* the topmost node contains 8 tweets, 3 Spam and 5 Not Spam, giving the proportions 32% Spam and 68% Not Spam. For each split of a node, the observations (rows) from Table 1 are partitioned. For example, in the left most node, which represents $Reported =$ No, the observations $e_5, e_7,$ and e_8 have been partitioned from the original dataset into a subset. This subset has the classifications 0% Spam and 100% Not Spam since each observation has $Class =$ Not Spam. In a given node, the classification with the higher proportion is the *classification decision* of the observations within the node. Thus, for the observations in the leftmost node the classification decision is Not Spam.

Figure 1: Sample decision tree using data from Table 1.



2.1 Decision Trees

The goal of a decision tree is to recursively partition the observations from a dataset until the feature of interest, T, can be classified with maximum probability. We examine how splits in the nodes, as seen in Figure 1, are decided.

Suppose that each observation is labeled as *pos* or *neg* by T. In Table 1, T is the *Class* feature and *pos* would refer to Spam and *neg* to Not Spam. We then want to quantify the average information content of each observation when it is labeled as *pos* or *neg*. Denote p_{pos} as the proportion of observations that are labeled as *pos* by T.

Suppose that $p_{pos} = 1$, *i.e.* every tweet is spam. Then, if we take a random observation and see that $T = pos$, we haven't learned anything new about the observation. This is because every observation will have $T = pos$; thus by looking at label given by T, no new information about the observation is learned. In the case that $p_{pos} = 1$ we say the average information content of the observations is 0 since T doesn't provide information about the observations.

Alternatively, if $p_{pos} = 0.5$ then *pos* and *neg* observations have the same probability of occurrence. Now, if we take a randomly drawn observation and see that $T = pos$ we know information about that specific observation since not all observations have $T = pos$. Additionally, as p_{pos} decreases and we see an observation with $T = pos$ the information increases because the observation is more unique than others among all observations. The standard way to quantify the average information content of the *pos* observations is:

$$I_{pos} = -\log_2 p_{pos}. \tag{1}$$

The negative sign compensates for the fact that logarithm will always be negative since $p_{pos} \in [0, 1]$. Additionally, as p_{pos} decreases, the quantity of information increase.

However, there are also observations with $T = neg$. In order to calculate the average information contents of T we must consider both *pos* and *neg* observations. To compute this, we introduce entropy [5]. Entropy is calculated as

$$H(T) = -p_{pos} \log_2 p_{pos} - p_{neg} \log_2 p_{neg}. \tag{2}$$

Note that $H(T)$ becomes maximized at 1, namely when $p_{pos} = p_{neg} = 0.5$. Further, entropy reaches a minimum when either $p_{pos} = 1$ or $p_{neg} = 1$. When either $p_{pos} = 1$ or $p_{neg} = 1$ it is known as *perfect regularity* since all observations have the same label of T. Alternatively, if $p_{pos} = p_{neg} = 0.5$ it is known as a *total lack of regularity*.

Given the data in Table 1, we can calculate $H(Class)$ as

$$H(Class) = -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right) = 0.954.$$

Note that since $H(Class)$ is close to 1 there is a lack of regularity.

Decision trees were invented to deal with a lack of regularity. Given data with a lack of regularity, we use features which contribute the most information to classifying the label of T. These features are used to partition the dataset into subsets with increased regularity. For example, of the features from Table 1 (excluding ID and $Class$) we want to identify which feature can best classify observations into $Class =$ Spam and $Class =$ Not Spam.

We will denote a feature of the dataset as X. Note that X will partition T into subsets, T_i, for each of the values that X can take. For our case, where $T = Class$, suppose $X = Account\ Age$ and recall that *Account Age* can take the values Old or New. In this case, $Class$ would be partitioned into $Class_{\text{Old}} = \{e_1, e_2, e_4, e_8\}$ and $Class_{\text{New}} = \{e_3, e_5, e_6, e_7\}$.

Let the number of elements inside of T_i be denoted as $|T_i|$. Then, the probability that a randomly drawn observation from the dataset be contained in T_i is

$$P_i = \frac{|T_i|}{|T|}.$$

Then, to calculate the entropy of T given the values of X we can sum across the entropies of all T_i, weighted by the respective P_i. The equation for this is given by the following:

$$H(T|X) = \sum_i P_i \cdot H(T_i). \tag{3}$$

$H(T|X)$ gives the entropy of a system where the labels of T, (*pos*, *neg*), and the values of X are known. Specifically, this will tell us the regularity of T when the values of X have been accounted for. Recall that we are trying to take a system with a lack of regularity and make it more regular so lower values of $H(T|X)$ are better.

We will continue our example with *AccountAge*. Using the data in Table 1, $|Class_{\text{Old}}| = |Class_{\text{New}}| = 4$. Thus, we get $P_{Old} = P_{New} = 0.5$. To calculate $H(Class|Account\ Age)$ we make the following computations:

$$H(Class_{\text{Old}}) = -\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right) = 0.881,$$
$$H(Class_{\text{New}}) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right) = 1,$$
$$H(Class|Account\ Age) = 0.5 \cdot 0.881 + 0.5 \cdot 1 = 0.9405.$$

Thus the entropy when both the *Class* label and the values for *Account Age* are known is 0.9405.

We then identify which feature supplies the most information about how to classify an observation into *pos* or *neg*. This is known as the *information gain* from each feature. Continuing to denote a feature as X, information gain is realized as the difference between the entropy (regularity) *before* X has been taken into account and *after* X has been taken into account, *i.e.*

$$I(T|X) = H(T) - H(T|X). \tag{4}$$

For *AccountAge* this is:

$$\begin{aligned} I(Class|Account\ Age) &= H(Class) - H(Class|Account\ Age) \\ &= 0.954 - 0.9405 = 0.014. \end{aligned}$$

Then, by applying this same process to each feature, we can find which feature gives the most information about classifying the labels of *Class* and thus serves as the best choice to split on within a decision tree.

Using Equation (3) we find that

$$H(Class|URL) = 0.9511,$$
$$H(Class|Reported) = 0.607.$$

Lastly, Equation (4) gives us the following:

$$\begin{aligned} I(Class|URL) &= H(Class) - H(Class|URL) \\ &= 0.954 - 0.9511 = 0.003, \\ I(Class|Reported) &= H(Class) - H(Class|Reported) \\ &= 0.954 - 0.607 = 0.347. \end{aligned}$$

Thus we conclude that the maximum amount of information is contributed by the *Reported* feature. Looking back at Figure 1, we see that this indeed was the first feature split on. We will then apply the same process on the 5 tweets on the right node after the split, but stop on the left node since there is perfect regularity, *i.e.* all the observation have the same *Class* value. In general, decision trees recursively apply this process to each node until either perfect regularity is attained or partitions fail to decrease the lack of regularity.

The procedure given above assumes all features have discrete values, *i.e.* the feature can be broken up into categories. Unfortunately, this does not match many sets of realistic data; many features, *e.g.* the time interval between two consecutive tweets, take *continuous values*. A continuous feature is one that can take an infinite number of values. Luckily, transforming continuous features into discrete features is relatively simple.

Consider a continuous feature denoted as x. We choose a threshold value, θ, of x such that if $x < \theta$ then the value is $true$ and otherwise $false$. If there are N observations in the dataset then we will consider $N - 1$ values for θ. Define a threshold as

$$\theta_i = \frac{x_i + x_{i+1}}{2}.$$

Suppose x takes on 4 different values, 3, 6, 8, and 10. Then we have $\theta_1 = 4.5$, $\theta_2 = 7$ and $\theta_3 = 9$. Then the amount of information for each threshold θ_i is calculated and the threshold that supplies the maximum information is chosen.

2.2 Random Forests

While decision trees are often a very useful machine learning technique they have their drawbacks. Decision trees will often build a tree to match a specific dataset and fail to extrapolate to other datasets containing the same features. This is known as *overfitting*. Overfitting leads to variance in the predictions of decision trees constructed on different subsets of the data. Random forests were developed to average many decision trees in order to minimize this variance.

To reduce variance, multiple decision trees are built using different subsets of the data and then their results are combined. Let B denote the number of trees to build. Then each tree is constructed on a sample of the dataset. Samples are done with replacement, *i.e.* sampled observations are put back into the dataset after a tree is built. This process of making many trees on samples of the dataset is known as *bagging* the dataset. When a new observation is to be classified it is put through each of the B trees and the observation's classification decision is determined according to the majority rule of the final classification decisions of the B trees. By bagging the dataset, there is less variance due to a decrease in the sensitivity to noise. However, it was discovered that if the trees are highly correlated with one another, the decrease in variance is likely minimal. [4]

In addition to bagging, random forests also employ *feature bagging* which avoids correlated trees. At each split in the decision tree, a random subset of features is considered as opposed to the entire set of features. It is common practice that when there are p features in the data set, $\sqrt{p}$ are randomly made available at each split. The intuition is if one or few features are very strong predictors of the feature of interest, T, then most of the B trees will contain that feature, thus correlating the trees. By avoiding correlated trees, the decrease in variance is maximized.[1]

2.3 Model Evaluation

Now, we will introduce methods of comparing models for random forests. In order to evaluate two random forests they must first be *trained* and then *tested*. This uses two distinct datasets: a *training set* and a *test set*. The training set is used to construct the random forest and should be balanced, *i.e.* the ratio of spam to not spam should be 1:1 [4]. Additionally, the training set is no part of model evaluation

and is used solely for the construction of the random forest as discussed in Sections 2.1, 2.2.

Once the model is built, it can be evaluated using the test set. For the test set, the *Class* feature is known but has been removed before being evaluated by the random forest. The random forest will evaluate an observation from the test set and give a classification, *i.e.* spam or not spam. Once the classification is given by the random forest we compare the predicted classification against the true value (which was removed prior to to classification). Doing so will result in four different categories: true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). A true positive would be an observation that a model predicted to be spam and whose *Class* label was also spam. A false positive would be an observation predicted to be spam by the model but whose *Class* label was not. True negatives and false negatives are analogous.

One method of evaluation is to compare the *accuracy* of the model. Accuracy indicates the ability of the model to correctly classify an observation. The calculation for Accuracy is given by

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{5}$$

Additionally, we calculate the proportion of positive classifications that were correct according to *Class*. This is known as the *precision* or *positive predictive value* of the model. Similarly, we calculate the proportion of positive classes that were classified as positive, known as the *recall* or *sensitivity*. The computation for each of these are:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{6}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{7}$$

Instead of directly comparing these terms between models, a combination of them, known as the *F-measure*, denoted as F, is formed. To compute F we take the harmonic mean of the precision and recall. Specifically,

$$F = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{8}$$

Note that $F \in [0, 1]$ since both precision and recall are proportions and can take values between 0 and 1. Using this formulation, F weights precision and recall equally and serves as a comparator to evaluate different models. Precision, recall, accuracy and F are evaluated for each study and presented in Section 4.

3. METHODS

The three studies considered use features to either identify a tweet as spam or a user as a spammer. Section 3.1 presents a study by Chen et al. that only considers easily computable user features and tweet features in order to have real-time classification of spam against not spam [2]. In contrast, in Section 3.2, we examine how Guo and Chen make use of geo-tagged data to help identify spammers and non-spammers. [3]. Section 3.3 presents the features constructed by Washha et al. to incorporate time-dependent features to identify spammers and non-spammers [6].

3.1 Tweet and User Content as a Feature for Spam Detection

The easiest features to extract are those that are included in the tweet itself and those tied to the account of the user who posted the tweet. The benefit of using the features directly connected to a specific tweet and user is the tweet can be processed in real time, *i.e.* the tweet can be recognized as spam or not spam almost immediately.

Chen et al. specifically chose features that could be computed as quickly as possible with the motivation that the longer spam exists, the more exposure it has to victimize users. They collected 12 of these computationally simplistic features, as listed in Table 2. Six of their features concern the user and the user history and the other six identify features about the specific tweet. These features could be taken directly from the data structure of the tweet.

Table 2: Features used by Chen et al. [2]. Features above the horizontal line indicate user features whereas features below are features of the tweet itself.

Features used by Chen et al.	
Feature	Description
accountAge	Age in days of an account since creation to the most recent tweet.
nFollower	Number of followers of a user.
nFollowing	Number of followees of a user.
nFavorite	Number of favorites a user received.
nLists	Number of lists a user has added.
nTweets	Number of tweets a user has sent.
nRetweets	Number of retweets a tweet has received.
nHashtag	Number of hashtags in a tweet.
nMentions	Number of users included in a tweet.
nUrls	Number of URLs included in a tweet.
nChar	Number of characters in a tweet.
nDigits	Number of digits in a tweet.

Chen et al. used the Twitter API to construct a dataset of 600 million tweets containing URLs. In doing this, they were able to classify tweets as spam or not spam based on if a tweet contained a malicious URL as listed by Trend Micro's Web Reputation Service. Approximately 1% (6.5 million) of all the tweets were classified as spam using this system.

Using this data, they constructed four test sets by having two different proportions of spam, 50% spam (test set I) and 5% spam (test set II) and two different sampling methods, one continuous and one random. Given the space limitation we discuss the two test sets in which continuous sampling was applied. Note that this study identifies spam on a tweet level rather than on a user level, which will differ from the other two studies presented.

Using these two test sets they tested their random forest using the features listed in Table 2. Results of these random forest are presented in Section 4.

3.2 Using geo-tags for Identifying Spam

Next, we consider the features one can build when using geographical location tags (geo-tags). Specifically, Guo and Chen constructed seven different geographical features to identify a user as a non-personal user [3]. Non-personal users include organization accounts and bots as well as spammers; for the sake of simplicity we will refer to non-personal users as spammers and personal users as non-spammers.

Many of their features use *tweeting speed* in their calculation. They defined tweeting speed as the geographical distance (in miles) between two consecutive tweets of a user divided by the time interval (in minutes) between the two tweets. For example, if a user, Dan, tweets from Clontarf, MN and 20 minutes later he again tweets from Morris, MN the tweeting speed would be 18.9 miles (the distance between Clontarf and Morris) divided by 20 minutes. This would result in a tweeting speed of 56.7 miles per hour. The intuition behind using tweeting speed for feature construction is spammers may change location faster than is possible given plausible transportation speeds, excluding air travel. The seven geographical features can be seen in Table 3.

Table 3: Geo-tagged features used by Guo and Chen [3]

Features used by Guo and Chen	
Feature	Description
maxSpeed	Maximum tweeting speed between two consecutive tweets for a user.
meanSpeed	Average tweeting speed for a user.
maxSpDist	Distance interval associated with the maximum tweeting speed.
spCntsPerM	Average monthly number of times a user has a tweeting speed higher than 90 miles per hour.
cntyChgsPerM	Number of times per month a user crosses county boundaries between consecutive tweets.
cntyCntsPerM	Average monthly number of counties that a user has been to.
locsCntsPerM	Average monthly number of mentioned/replied users who have geo-tagged tweets.

Using the Twitter streaming API, Guo and Chen collected over 600 million tweets from a little over 5 million unique users. In total, 2 million users had at least 5 geo-tagged tweets. They performed their feature extraction on users who had at least 5 geo-tagged tweets. To create a training dataset random sampling was applied and manual classification of spammer and non-spammer was determined. Similar methodology was applied to make a test set of 1,177 users: 925 non-spammers and 252 spammers. The results from Guo and Chen's analysis are presented in Section 4.

3.3 Time as a Feature for Identifying Spam

Next, we study how to leverage time to identify spammers. The intuition is that it is arduous to alter time-dependent features of a tweet. Washha et al. placed features into one of the following categories: user features, content features, posting diversity features, and similarity features. [6]

3.3.1 Time-Dependent User Features

Followers and Followees - When a user has a low number of followers, a very small ratio of followees to followers, or a very small number of bi-directional relationships then they have a high probability of being a spammer. To counteract this, spammers create many accounts such that the accounts follow each other thus increasing followers, bi-directional relationships, and evening out the ratio of followees to followers. However, this means that the majority of these accounts will be made in the same time period.

Three features are created to exploit this occurrence. Given a user, u, and a set of $users$ (*e.g.* followers, followees), the mean and variance of the account age difference between u and $users$ is computed. The intuition is if the mean or variance of difference in account age is very small, then the user, u, has high likelihood of being a spammer. Lastly, a final computation is made by giving account age as a weighting factor such that older accounts are given larger weights. Again, a low value of this feature implies that u has a high likelihood of being a spammer. These features are calculated for three different sets of $users$:

- *followers* - followers of a user u
- *followees* - the followees of user u who have not been verified by Twitter
- *bi-directionals* - those who are a followee of u and are also followed by u

Thus a total of nine features are created, three for each of the three sets of $users$.

Profile Description - Twitter accounts include profile description which can consist of up to three objects: words, hashtags, and URLs. Spammers will often defer from using spam words within the profile description as to avoid detection. More so, a spammer will fill out the profile description to appear as a normal user. Still, it is likely that each of a spammers' accounts have the same or similar profile elements and that the accounts are created in the same time period. Washha et al. capture this weakness by calculating the similarity between a user's profile and their followers and followees. The similarity calculation was obtained by using the Kullback-Leibler Divergence and the Jaccard Similarity, details of which can be found in the study by Washha et al. Since spammers' accounts are made around the same time, Washha et al. used the difference between u's age and the ages of u's followers and followees as a weighting factor. [6]

3.3.2 Content Across Time Features

Posting Behavior - Spammers often display similar, systematic posting behaviors while real users display randomness in their posting behavior. Washha et al. employs the use of auto correlation between three different attributes: hashtags, mentions, and URLs. While the specific statistical techniques are not discussed in this paper, they are available in the study conducted by Washha et al. [6]

3.3.3 Posting Diversity Features

Real users and spammers can intensively use the same hashtags, URLs, and user mentions. When basic features are employed (number of URLs, number of hashtags, number of user mentions, etc.), the model often fails to distinguish a spammer from a real user. To combat this, a feature is constructed that measures the diversity of an instance set, I_s. For example, one instance set may be for hashtags, $I_{hashtag} = \{\#H_1, \#H_2, \#H_3\}$. The posting diversity (PD) of an instance set, I_s, and the user u is computed as

$$PD(u, I_s) = \frac{|I_s|}{|u.Tweets|},$$

where $|I_s|$ is the number or elements in I_s and $u.Tweets$ is all of a user's tweets.

A value of 0 for PD indicates that the instance set is empty, whereas a value of 1 means that each element in I_s is used only one time in the user u's tweets. This function is then applied to hashtags, mentions, URLs, and textual words (words that are not URLs, mentions, or hashtags).

3.3.4 Tweet Similarity Across time Features

Spammers will often tweet the same tweet many times in a row. Tweet similarity can be used to help identify a spammer as opposed to a real user. A tweet similarity function, $T_{sim}(t_1, t_2)$, is constructed where t_1 and t_2 are two tweets such that $t_1.Time > t_2.Time$. Note that $t_1.Time$ identifies the time at which t_1 was posted. T_{sim} is used to compare the similarities of two tweets using only textual words. The use of strictly textual words avoids potential similarities due to hashtags and URLs. Additionally, spammers often post similar tweets within a short period of time, whereas a real user may post the similar tweets but it will typically be after a longer period of time. Washha et al. controls for this by creating a time weighted tweets similarity, $TWTS(u)$.

$TWTS(u)$ is calculated as

$$TWTS(u) = \frac{\sum_{t1} \sum_{t2} T_{sim}(t1, t2)}{\frac{|u.Tweets| * (|u.Tweets| - 1)}{2}}.$$

$TWTS$ ranges from 0 to 1 where a value of 0 means there is no similarity between the users tweets (or the user has never posted a tweet) and a value of 1 means that all tweets are duplicates of one another within a relatively short time.

3.3.5 Data Construction

Washha et al. used the Twitter API to construct a testing set of 1,082 spammers and 1,221 non-spammers. Roughly 300,000 tweets from 7,189 unique users were collected total. From there, they used a manual process to classify each user as a spammer or non-spammer. They used a three step process. First, if the account had been banned at the time of the manual process, the account was labeled as a spammer. Next, if the account had been verified by Twitter it was then labeled as a non-spammer. The rest of the accounts were manually inspected to determine whether the account was a spammer or non-spammer.

4. RESULTS

The model results from each of the three studies are presented in Table 4. Precision (p), recall (r), F-measure (F), and accuracy are presented as defined in Section 2.3. Between the two test sets of Chen et al., there was a sharp drop in F when the proportion of spam in the test set dropped. Upon further analysis it was discovered that when the total spam tweets decreased, the number of false positives rose drastically, lowering the precision of the random forest [2].

Each model displayed high accuracies (≥ 0.931). Excluding test set II by Chen et al., all values of F are above 0.93, showing that the models have high precision and recall.

Direct comparisons between models should be taken with reservation; each study measures something different. Chen et al. looked at individual tweets, Guo and Chen studied non-personal users, and Washha et al. studied spammers. That being stated, each study performs similar in terms of accuracy and F, excluding test set II from Chen et al.

Table 4: Model results from the 3 analyses studied: Chen et al., Guo and Chen, and Washha et al..

Model Results Of The 3 Studies					
Study	% Spam	p	r	F	Accuracy
Chen et al.: I	50.0%	0.929	0.943	0.936	0.936
Chen et al.: II	5.0%	0.407	0.929	0.566	0.978
Guo and Chen	21.4%	0.959	0.959	0.958	0.959
Washha et al.	46.9%	0.932	0.931	0.931	0.931

5. CONCLUSIONS

This paper summarized three unique ways of classifying spam tweets, spammers, and non-personal users using different features. All models displayed high accuracies and all but test set II of Chen et al. showed high F-measures, indicating a high classification ability of the different models.

Chen et al. performed a novel analysis when noting that typically only 5% of tweets are spam and adjusting for this by creating a separate test set. Future work could reapply this notion to the work of Guo and Chen as well as that of Washha et al. to see how this change affects their analysis.

All three models showed that random forests, in combination with particular features, can be used as strong classifiers of spam, spammers, and non-personal users. In addition, applications of random forests may have unexpected results when applied to real data if a realistic proportion of spam/spammers is not accounted for in the test set.

Acknowledgments

I thank my adviser, Peter Dolan, and my senior seminar professor, Elena Machkasova, for their invaluable advice and comments throughout the research and writing process. Additionally, I thank my alumni reviewer, Jacob Opdahl, for his revisions and comments during the review process.

6. REFERENCES

[1] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] C. Chen, J. Zhang, X. Chen, Y. Xiang, and W. Zhou. 6 million spam tweets: A large ground truth for timely twitter spam detection. In *2015 IEEE International Conference on Communications (ICC)*, pages 7065–7070. IEEE, 2015.

[3] D. Guo and C. Chen. Detecting non-personal and spam users on geo-tagged twitter network. *Transactions in GIS*, 18(3):370–384, 2014.

[4] M. Kubat. *An Introduction to Machine Learning*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[5] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[6] M. Washha, A. Qaroush, and F. Sedes. Leveraging time for spammers detection on twitter. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems*, MEDES, pages 109–116, New York, NY, USA, 2016. ACM.