Information Sciences 260 (2014) 64–73

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Twitter spammer detection using data stream clustering

Zachary Miller[a], Brian Dickinson[a], William Deitrick[a], Wei Hu[a,*], Alex Hai Wang[b]

[a] Department of Computer Science, Houghton College, Houghton, NY, United States
[b] College of Information Sciences and Technology, The Pennsylvania State University, Dunmore, PA, United States

A R T I C L E   I N F O

Article history:
Received 8 August 2012
Received in revised form 9 October 2013
Accepted 1 November 2013
Available online 17 November 2013

Keywords:
Twitter
Spam detection
Clustering
Data stream

A B S T R A C T

The rapid growth of Twitter has triggered a dramatic increase in spam volume and sophistication. The abuse of certain Twitter components such as “hashtags”, “mentions”, and shortened URLs enables spammers to operate efficiently. These same features, however, may be a key factor in identifying new spam accounts as shown in previous studies. Our study provides three novel contributions. Firstly, previous studies have approached spam detection as a classification problem, whereas we view it as an anomaly detection problem. Secondly, 95 one-gram features from tweet text were introduced alongside the user information analyzed in previous studies. Finally, to effectively handle the streaming nature of tweets, two stream clustering algorithms, StreamKM++ and DenStream, were modified to facilitate spam identification. Both algorithms clustered normal Twitter users, treating outliers as spammers. Each of these algorithms performed well individually, with StreamKM++ achieving 99% recall and a 6.4% false positive rate; and DenStream producing 99% recall and a 2.8% false positive rate. When used in conjunction, these algorithms reached 100% recall and a 2.2% false positive rate, meaning that our system was able to identify 100% of the spammers in our test while incorrectly detecting only 2.2% of normal users as spammers.

1. Introduction

Social networking sites have exhibited exponential growth over the last few years. Among the most popular sites are MySpace, Facebook, and most recently Twitter. While the popularity of MySpace is declining and Facebook’s membership has plateaued at around 130 million, Twitter is still growing with around 25 million active users. Facebook’s slowing expansion due to market saturation stands in contrast to Twitter’s growth rate of about 30% annually [9].

Unfortunately, the proliferation of social networking has contributed to an increase in spam activity [15]. Spammers send unsolicited messages to users with varying purposes, which include, but are not limited to advertising, propagating pornography, phishing, spreading viruses, and degrading Twitter’s reputation [4]. “Hashtags”, “mentions”, and shortened URLs are frequently abused by spammers, making them indicators that a tweet may be from a spam user. Mentions are normally used to reply to another user or to send a message, which will appear on his profile. A spammer may exploit this by mentioning other users in order to spread spam beyond his limited following. Hashtags are used to denote trending topics making them more searchable for users and are abused by spammers, who attach the most popular hashtags to messages with links to unrelated topics. URL attacks are aided by Twitter’s 140 character limit on tweets as many legitimate users need to use link-shortening services to reduce the length of their URLs. This allows spammers to deceive users as the shortened links do not appear to be malicious, which is particularly important because links are a necessary part of almost any spamming attack. On Twitter, links are especially dangerous because it is very difficult to determine their destination without following

* Corresponding author. Tel.: +1 5855679612.
E-mail address: wei.hu@houghton.edu (W. Hu).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.11.016

them to potentially harmful sites [14]. The ability to disguise URL destinations has made Twitter a particularly attractive target for spammers, which has motivated the development of several spam detection techniques.

In the first study focusing on Twitter spam detection, a data set of approximately 25,000 Twitter accounts was collected over several weeks with a web-crawler using Twitter's API [15]. A variety of predictors were then used for classification using 10-fold cross validation with a set of user and content based features. The users classified as spammers by the algorithms were then manually checked to determine the false positive rate. The Naïve Bayes Algorithm provided the best results with 91.7% precision, recall, and F-measure.

Researchers in [4] collected an enormous data set comprised of every known active public account on Twitter as of August 2009. This data set included millions active and inactive users and 1.7 billion tweets. Tweets containing one of the three most popular hashtags (#musicmonday, #MichaelJackson, and #SusanBoyle) were then identified and removed for manual labeling. A Support Vector Machine running 5-fold cross validation was used on this subset to evaluate the effectiveness of a similar feature set to [15]. They were able to identify 70% of spammers while maintaining a relatively low false positive rate of about 4%.

One of the most recent studies [13] used a collection of learning algorithms including Random Forest, Support Vector Machine, K-Nearest Neighbor, and Naïve Bayes. Each was used to evaluate a labeled data set of approximately 1000 Twitter accounts utilizing a similar feature set to the other previously mentioned studies [4,15]. The Random Forest algorithm produced the top results with over 95% precision, recall and F-measure. SVM performed nearly as well, achieving around 93% in each category.

In each of these previous studies, classification was utilized for spam detection on Twitter. The aim of this study is to develop an anomaly detection system for identifying spammers on Twitter using account information and streaming tweets. Typically binary classification applies to a data set that has balanced class labels, while anomaly detection is effective when the majority of a data set is one class; data outside that class are outliers. Due to the speed and volume of Twitter traffic, stream mining is the most natural technique for spam detection. Data streams are characterized by large volumes of continuous data evolving over time. Because of this, stream mining must make use of only one pass over the data. Therefore it is preferable for use in the Twitter environment where millions of tweets are posted every day.

2. Materials and methods

2.1. Data and their representation

The data set for this study included 3239 user accounts with a sample tweet from each account. Among the 3239 users, 208 were spam accounts from [15]. Because an estimated 6% of all Twitter accounts are spammers, our 208 spam users were combined with 3031 randomly selected verified normal users to form our complete data set. The 3031 normal users were randomly selected from a set of 37,000 accounts that tweeted during a one-hour period on March 3rd 2012 using the Twitter Streaming API. Each instance, consisting of the collected user account information and a sample tweet, was manually labeled as spam or non-spam in order to enable precise evaluation of our spam detection algorithms' performance. This process takes a great deal of time due to hourly limitations on requests made to Twitter. While studies such as [15,4] have opted for a larger data set with incomplete results, studies like [13] have chosen smaller data sets to facilitate more thorough analysis. Because we wanted more conclusive results, we elected to use a smaller data set that could be manually labeled.

After being labeled, the instances were divided into two separate sets: training and testing. Each set contained approximately 1500 normal and 100 spam users, which reflects the ratio of spam to non-spam users on Twitter (Table 1). The training set was used to tune the parameters of the algorithms that would be used in the final tests. Each instance was then parsed and represented by a vector of 107 numerical feature values.

The features used to represent each instance fall into three principal categories: content, user information, and tweet text (Table 2). Content-based features include link, mention, and hashtag counts denoted by "http", "@", and "#" respectively. Links are a necessary tool for distributing spam content and as such are included in nearly every spam tweet. The presence of hashtags and mentions in a large percentage of tweets is indicative of a spammer [15] because these tools increase audience size. User-based features include the number of followers, the number of accounts followed, and the follower ratio.

Table 1
Training and testing set data.

	Label	Number of users
Training	Spam	104
	Non-spam	1483
	Total	1587
Testing	Spam	104
	Non-spam	1548
	Total	1652

Table 2
Feature list.

Feature number	Feature description
F1	Follower count
F2	Friend count
F3	Favorites count
F4	Listed count
F5	Tweet count
F6	Re-tweet count
F7	User verified
F8	Age day
F9	Follower ratio
F10	Link count
F11	Reply/mention count
F12	Hashtag count
F13–107	One-gram characters

These three features all relate to the tendency of spammers to follow large numbers of users with a low percentage following them back. Other important user statistics include favorites, account age, the number of lists on the account, and re-tweet count. Information such as account age and tweet count is informative because of the limited life-span of spam accounts on Twitter. These features are listed in Table 2, where features F1, F2, F9, F10, F11, and F12 were used in [15], and the remaining were novel features proposed in this study. Tweet text-based features, introduced in this study, are based on a simple count of 95 one-gram characters. This 95 character subset was chosen from the complete ASCII set for accessibility on a standard US keyboard. The combination of the previous 12 features with these 95 one-grams forms a total of 107 features.

2.2. Methods

To detect spam on Twitter, modified DenStream and StreamKM++ stream clustering algorithms are proposed. The novelty of our approach is to treat spam identification as an anomaly detection problem rather than as a classification problem. In this anomaly detection approach, a clustering model is built on normal twitter users with all outliers being treated as spammers. In this manuscript we use the terms instance and point interchangeably in the description of our methods, since in some contexts one is clearer than the other.

2.2.1. Density-based clustering

Density-based clustering does not require the input of a predefined number of clusters, and can form clusters with arbitrary shapes. Some examples of density-based clustering algorithms are DBSCAN [8], OPTICS [2], and PreDeCon [6,11]. This type of clustering is traditionally done in batch learning environments, but recent research in the area of stream data mining has led to the development of multiple stream density-based clustering algorithms. One such stream data mining algorithm is DenStream, which applies a damped window to DBSCAN [7].

DBSCAN: DBSCAN [8] is a popular density-based clustering algorithm with two simple parameters, ε and minimum points. The ε parameter is used to define the ε -neighborhood of a point as all of the points that are within a distance of ε of the first point. More formally, an ε-neighborhood of a point p in the data set D is mathematically represented as:

$$N_\epsilon(p) = \{q \in D : \quad dist(p,q) \leqslant \epsilon\},$$

where $dist(p,q)$ is the Euclidean distance between the points p and q. Once DBSCAN calculates the ε -neighborhood for a single point, it considers the neighborhood as a core-object if there are more points than the minimum points parameter within the neighborhood. If a core-object is found, the clustering expansion proceeds with the points in the ε -neighborhood of the core-object. Once a core-object which represents a cluster is found, DBSCAN randomly selects another point and continues until all other points have been selected. As one of the first density-based clustering algorithms, DBSCAN has served as the basis for many different density-based clustering algorithms.

DenStream: DBSCAN, while simple and effective, is not designed to handle dynamic data streams. By extending the concept of core-objects to core-micro-clusters, DenStream [7] is able to cluster in the data stream environment. Micro-clusters are compact representations of large sets of dense data weighted by a fading function, which is defined as:

$$f(t) = 2^{-\lambda t},$$

where $\lambda > 0$ is the decay factor and t is the current time-step. Core-micro-clusters further differ from the traditional core-objects of DBSCAN with the inclusion of three new attributes, weight, center, and radius. These attributes are used in the clustering as the radius of a core-micro-cluster must be less than ε and its weight must be greater than another user-defined parameter μ. The weight, center, and radius can be calculated for a set of close points $p_1, p_2, p_3, \ldots, p_n$ at time t, with time-stamps $T_1, T_2, T_3, \ldots, T_n$ using the following formulas:

$$w = \sum_{i=1}^{n} f(t - T_i).$$

$$c = \frac{\sum_{i=1}^{n} f(t - T_i) p_i}{w}.$$

$$r = \frac{\sum_{i=1}^{n} f(t - T_i) dist(p_i, c)}{w}.$$

The core-micro-cluster adds a decay factor to the core-objects of DBSCAN, but is unable to be updated as new data points arrive. To introduce this functionality into DenStream, a potential core-micro-cluster or p-micro-cluster is employed. p-Micro-clusters use the weighted linear and squared sums, $\overline{CF^1}$ and $\overline{CF^2}$, to determine the center and radius values. $\overline{CF^1}$ and $\overline{CF^2}$ are calculated using the formulas

$$\overline{CF^1} = \sum_{i=1}^{n} f(t - T_i) p_i,$$

$$\overline{CF^2} = \sum_{i=1}^{n} f(t - T_i) p_i^2,$$

which change the formulas for the center and radius to be

$$c = \frac{\overline{CF^1}}{w},$$

and

$$r = \sqrt{\frac{\overline{CF^2}}{w} - \left(\frac{\overline{CF^1}}{w}\right)^2}.$$

Although p-micro-clusters allow DenStream to cluster incoming dense points, one more data structure is needed to handle dynamic data streams. An outlier-buffer is used to temporarily hold any data points that are not considered part of p-micro-clusters. This buffer maintains these points until they create a dense micro-cluster and are able to become a p-micro-cluster. These outlier-micro-clusters or o-micro-clusters are necessary in case the data stream changes significantly over time.

Using these concepts, DenStream initially receives a set of points, on which DBSCAN is run to create an initial set of p-micro-clusters. Once the initial p-micro-clusters are formed, DenStream waits until a new point arrives from the stream. This new point p first attempts to join its nearest p-micro-cluster. If the point falls within the radius of the nearest p-micro-cluster, it is merged and the radius, center and weight of the cluster are updated. If p fails to merge with a p-micro-cluster, it then attempts to merge with an existing o-micro-cluster in the outlier buffer. If the new o-micro-cluster now has a weight large enough, it becomes a p-micro-cluster and is removed from the outlier buffer. If the point p is unable to merge with any other micro-cluster, DenStream creates an o-micro-cluster on p and places in the outlier buffer. After the new point has arrived, DenStream will periodically check and remove any o-micro-clusters which have a weight that falls beneath the lower weight bound. This bound is calculated using the formula

$$\xi(t_c, t_o) = \frac{2^{-\lambda(t_c - t_o + T_p)} - 1}{2^{-\lambda T_p} - 1},$$

where t_c is the current time and t_0 is the creation time of the o-micro-cluster.

In this study we view spam detection on Twitter as an anomaly detection problem, therefore DenStream was modified by removing certain functions so that the normal models made of p-micro-clusters would not be tainted by instances classified as spam. This change will likely leave out a few normal instances from the clustering, but it safeguards the accuracy of the predictions as the streaming nature of DenStream will adapt to gradual changes of normal behavior. The p-micro-clusters in DenStream are used to model normal instances. If any incoming point is not merged with a p-micro-cluster, it is classified as abnormal and placed in the outlier buffer until deletion. By removing the function for an o-micro-cluster to become a p-micro-cluster, this modified DenStream can only cluster instances that are considered normal. The overall modification is displayed in Algorithm 1.

```
Algorithm 1: Modified DenStream
  • Initial Phase:
    ○ Run DBSCAN on first 300 points to generate a normal model of p-micro-clusters
  • Online Phase: For each point p that arrives from the stream
```

(continued on next page)

```
    ○ Attempt to merge p with nearest p-micro-cluster
      ■ If successful, consider as normal
      ■ Else, consider as spam
```

2.2.2. k-Means based clustering

In contrast to density-based clustering algorithms, k-means based clustering algorithms form spherical clusters using Euclidean distance. StreamKM++ was chosen from among these for spam detection because of its speed and effectiveness.

k-Means Clustering: k-Means [12] is a well-known clustering algorithm which finds a predefined number of clusters *k*. In the first step of k-means, *k* points are randomly selected as centers and used to cluster all remaining points. Once every other point is associated with a center, the mean for every cluster is calculated and named the new center. This process continues until the centers no longer move [12,16]. Although k-means is able to effectively cluster similar points together through the Euclidean distance, the user needs to know how many clusters should be found. If an incorrect *k* value is chosen, the clustering can be inaccurate, and multiple selections for *k* are often recommended to find the best model [16].

One of the biggest shortcomings of the k-means algorithm is the random selection of initial centers. If k-means selects two centers that are very close to each other, the resulting clusters have the potential to be inaccurate. To minimize these types of errors, Arthur and Vassilvitskii [3] proposed the k-means++ algorithm, which first chooses one initial center with a uniform probability. Then k-means++ chooses the next center by calculating new probabilities for every other point using the formula $\frac{dist(x,c)^2}{\sum_{x\in X} dist(x,c)^2}$, where $dist(x,c)$ is the shortest distance from a point x in data set X to its closest center c already chosen. By doing this, the starting centers will most likely be far enough apart to allow the original k-means algorithm to run successfully.

k-Means++ improves the clustering performance of k-means, but is designed to process batch data. To add the ability for k-means++ to update its clusters as new points arrive from a data stream, StreamKM++ was developed [1]. StreamKM++ introduces the concept of a coreset which is defined as a weighted point, a subset of the input. Clusters built on these coresets are a good approximation of the clustering of the original points [10]. To begin clustering, StreamKM++ first extracts a small set of points from the data stream and uses a merge and reduce scheme to keep the coresets manageable [10]. The number of coresets is a user-defined parameter m recommended as $m = 200 * k$ [1]. Once the number of coresets is determined, k-means++ clustering can be performed using the coresets as points. If a clustering model needs to be generated, StreamKM++ algorithm runs k-means++ five independent times and the tightest clustering result is chosen. The implementation of StreamKM++ in MOA [5] generates clusters whenever the number of points fills a sliding window of user-defined size. By implementing coresets and multiple runs of k-means++, StreamKM++ is able to effectively cluster streaming data.

To apply k-means-based clustering to spam detection on Twitter, some modifications were made to the StreamKM++ algorithm [5]. First, an additional parameter ε was added as the threshold for a point to be considered part of a cluster found using k-means++. Second, a detection scheme was implemented in order to classify points the moment they arrive from the stream. As each new point comes in, our modified StreamKM++ finds the nearest coreset cluster based on the most recent set of centers and calculates the Euclidean distance. If the distance is above the value of ε, then the point is detected as spam and is not inserted into the nearest coreset. Using this technique, spam instances are prevented from entering into the coresets. The modified StreamKM++ is further defined in Algorithm 2.

```
Algorithm 2: Modified StreamKM++
  • Initial Phase:
    ○ Generate a k-means++ clustering model based on the coresets of the first 300 normal points
  • Online Phase: For each point p that arrives from the stream
    ○ If p fills the sliding window
      ■ Run k-means++ to generate a new model
    ○ Calculate distance from p to nearest cluster
      ■ If the distance is less than $\varepsilon$,
        • p is considered normal and merged with coreset
      ■ Else
        • p is considered spam and removed
```

2.2.3. StreamKM++ and DenStream combined

We combined the two algorithms DenStream and StreamKM++ to further improve the spam detection while maintaining a low false positive rate. StreamKM++ was observed in our experiment to be better suited for detecting all spam instances without regard for incorrectly classifying normal ones. For this reason, we decided to choose three ε values that have near perfect detection but with a high, medium, and low false positive rate. This allows StreamKM++ to guarantee that predicted normal instances are in fact normal. The StreamKM++ component then outputs all predicted classes to be used by DenStream.

After StreamKM++ has determined its true negatives, DenStream assumes that every StreamKM++ predicted normal instance is correct, which lowers the false positive rate of DenStream. By carefully choosing a moderate value for ε, DenStream also corrects many of the false positive mistakes made by StreamKM++ and makes the final predictions (Algorithm 3).

```
Algorithm 3: StreamKM++ and DenStream combined
  • Run modified StreamKM++ algorithm to classify with zero false negatives.
    ○ Output predicted class values
  • Run modified DenStream using the instances with predicted classes from StreamKM++.
    ○ Assume predicted normal instances are true negatives
    ○ Output spam prediction results
```

To measure the performance of our spam detection systems, we employed the use of many commonly used metrics. If an instance is considered spam by the algorithm, it is either a True Positive (TP) or False Positive (FP) depending on if the prediction is correct. Similarly if an instance is correctly clustered into the normal model it is a True Negative (TN), otherwise it is a False Negative (FN). All four of these counts are used to determine the detection rate (recall), false positive rate, and the F-Measure.

2.3. Statistical significance of predictions

In order to determine the statistical significance of our predictions we demonstrate that it is very improbable that they are the result of random prediction. In the testing data set, the order of the initial 300 training instances was changed to randomize the initial model. The labels of the remaining instances were shuffled keeping the same distribution of the original instances. As a result some spam instances would be labeled normal and some normal instances labeled spam. This procedure was repeated to generate 1000 files. We performed the tests on each of these files in addition to the original file

Table 3
Comparison of best detection results from individual algorithms (p values are in the subscript).

	ε	Spec.	FPR	Accuracy	Bal. acc.	Prec.	F-measure	Recall
StreamKM++	1.20	$.9358_{p=0}$	$.0641_{p=0}$	$.9400_{p=0}$	$.9631_{p=0}$	$.5628_{p=0}$	$.7177_{p=0}$	$.9903_{p=0}$
DenStream	1.13	$.9719_{p=0}$	$.0280_{p=0}$	$.9731_{p=0}$	$.9812_{p=0}$	$.7055_{p=0}$	$.8240_{p=0}$	$.9904_{p=0}$

Table 4
Baseline detection performance of combined approach (p values are in the subscript).

	ε	Spec.	FPR	Accuracy	Bal. acc.	Prec.	F-measure	Recall
StreamKM++	.50	$.7001_{p=.586}$	$.3000_{p=.586}$	$.7232_{p=0}$	$.8500_{p=0}$	$.2176_{p=0}$	$.3574_{p=0}$	$1_{p=0}$
Combined	1.13	$.9687_{p=0}$	$.0313_{p=0}$	$.9711_{p=0}$	$.9844_{p=0}$	$.7272_{p=0}$	$.8421_{p=0}$	$1_{p=0}$

Table 5
Comparison of detection performance with and without one-gram features (p values are in the subscript).

	Algorithm	ε	Spec.	FPR	Accuracy	Bal. acc.	Prec.	F-measure	Recall
With one-gram	DenStream	1.13	$.9719_{p=0}$	$.0280_{p=0}$	$.9731_{p=0}$	$.9812_{p=0}$	$.7055_{p=0}$	$.8240_{p=0}$	$.9904_{p=0}$
	StreamKM++	1.10	$.9358_{p=0}$	$.0641_{p=0}$	$.9400_{p=0}$	$.9631_{p=0}$	$.5628_{p=0}$	$.7177_{p=0}$	$.9903_{p=0}$
Without one-gram	DenStream	1.13	$.9584_{p=.115}$	$.0416_{p=.115}$	$.9612_{p=0}$	$.9792_{p=0}$	$.6303_{p=0}$	$.7732_{p=0}$	$1.000_{p=0}$
	StreamKM++	1.10	$.8917_{p=0}$	$.1083_{p=0}$	$.8860_{p=0}$	$.8545_{p=0}$	$.3864_{p=0}$	$.5247_{p=0}$	$.8173_{p=0}$

Table 6
Comparison of algorithm detection performance metrics (p values are in the subscript).

	ε	Spec.	FPR	Accuracy	Bal. acc.	Prec.	F-measure	Recall
StreamKM++	1.20	$.9342_{p=0}$	$.0658_{p=0}$	$.9393_{p=0}$	$.9671_{p=0}$	$.5591_{p=0}$	$.7172_{p=0}$	$1_{p=0}$
DenStream	1.13	$.9687_{p=0}$	$.0313_{p=0}$	$.9711_{p=0}$	$.9844_{p=0}$	$.7272_{p=0}$	$.8421_{p=0}$	$1_{p=0}$
Combined	1.20 and 1.13	$.9783_{p=0}$	$.0217_{p=0}$	$.9800_{p=0}$	$.9892_{p=0}$	$.7939_{p=0}$	$.8851_{p=0}$	$1_{p=0}$

using the same test parameters. The p values reported in Tables 3–6 are the percentage of randomized predictions which outperformed our reported results.

3. Results

In this section, we report the spam detection performance of the each individual modified stream clustering algorithm. The performance of these systems is measured by a variety of metrics including recall, false positive rate, and F-Measure. Recall is the percentage of spam users detected and false positive rate is the number of users improperly considered as spam. F-Measure is used in several previous studies in spam detection on Twitter as an overall assessment of performance. Each algorithm, measured by these metrics, demonstrates its own spam detection capability. Therefore when combined, they

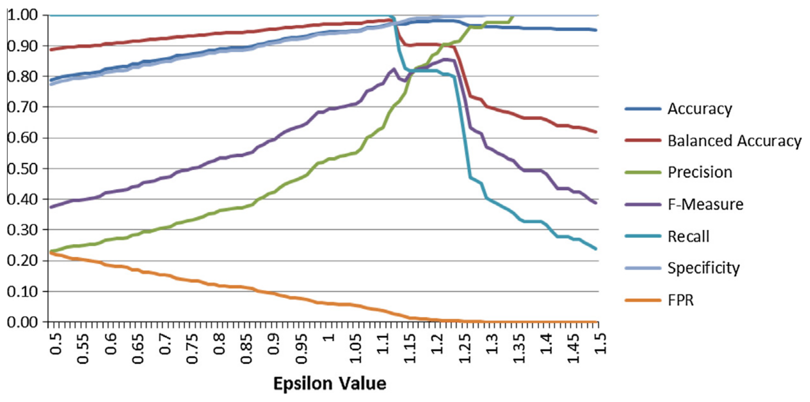

Fig. 1. DenStream spam detection performance metrics by epsilon value.

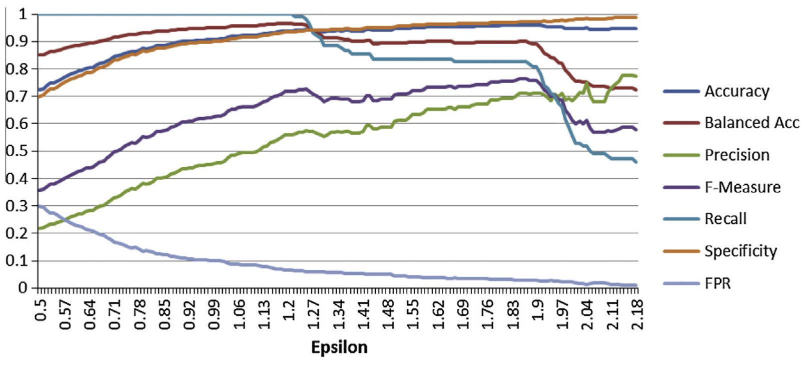

Fig. 2. StreamKM++ spam detection performance metrics by epsilon value.

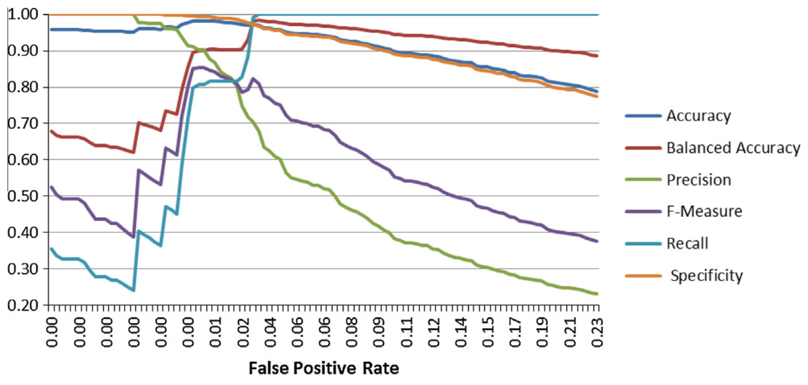

Fig. 3. DenStream spam detection performance metrics in relation with false positive rate.

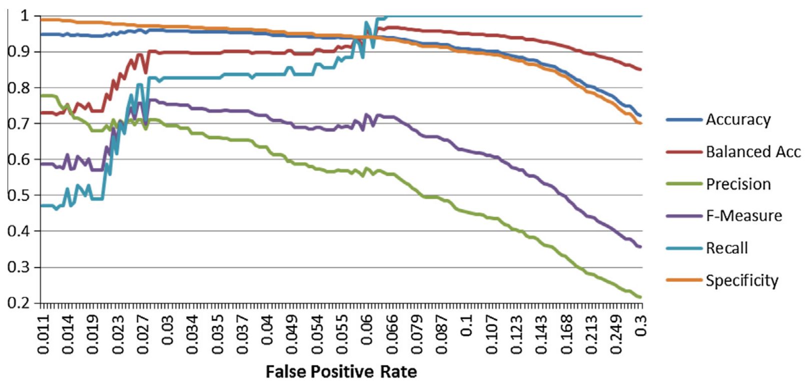

Fig. 4. StreamKM++ spam detection performance metrics in relation with false positive rate.

compensate for each other's relative weaknesses and improve detection. In addition to the result tables, graphs are included to demonstrate the performance of these algorithms over a broader spectrum based on their epsilon values and false positive rates (Figs. 1–4).

3.1. Spam detection with DenStream

Using our set of features, we attempted to cluster normal instances, treating outliers as spam. In order to achieve optimal performance, several parameters of DenStream, including initial points, minimum points, ε and λ, had to be tuned using the training set. Initial points refers to the number of points used in the initial DBSCAN clustering in DenStream, which was in this case 300 instances. Minimum points determines the number of points necessary to form a cluster; in our case this value is three. The decay factor was problematic because it eliminated the core cluster of normal instances before later instances could be added; for this reason we set λ to 0. ε was observed to be the most important parameter and underwent the most extensive tuning. ε values were tried in training between .5 and 1.5 in increments of .01. This determined the optimal ε value of 1.13 that was used later on the testing set. DenStream achieved an accuracy of 97.2%, precision of 70.6%, F-Measure of 82.4%, and amazingly, the recall and false positive rate were 99% and 2.8% respectively (Table 3).

3.2. Spam detection with StreamKM++

To choose a valid set of parameter values for testing, we ran StreamKM++ clustering algorithms multiple times in the training phase, during which StreamKM++'s ε value showed the greatest impact on spam detection. We observed through multiple trials that all normal instances unexpectedly clustered into three groups, which means that the clustering benefits from having $k = 3$. To select the number of coresets, the recommended $m = 200 * k$ was first used but resulted in inaccurate clustering. After this we decided to set $m = 60$ so that each coreset would be representative of more instances. Because we used 300 normal instances to generate a model for normal users for both the training and testing data sets, the sliding window size was set to this value. This means that after every 300 instances arrive on the stream, the k-means++ algorithm is run. After running multiple trials by varying the ε value with the training set, we found that ε is optimal between 1.1 and 1.21. When tested, the StreamKM++ algorithm is able to achieve roughly 93% accuracy with perfect recall and a false positive rate of 8.5%. Because a lower false positive rate is desirable, we further increased the ε value until our false positive rate was below 1% or until the recall drops below 70%. StreamKM++ was able to maintain a 70% recall rate with just over 2% false positives (Table 3).

3.3. Effect of one-gram features on spam detection

Previous studies mainly used user profile information for classification of spammers on Twitter. In this work we included additional information extracted from tweet text in the form of one-grams, which improved our detection results of both algorithms (Table 5). In particular, StreamKM++ increased its recall from 82% to 99% and lowered its false positive rate from 11% to 6%.

3.4. Effect of epsilon values on spam detection

To gain a better understanding of the impact of epsilon values on spam detection, we created graphs which plot the performance metrics (Figs. 1 and 2). Of note is that for epsilon values between 1.15 and 1.25, DenStream was able to maintain all metrics except for false positive rate above 80%. StreamKM++ did not achieve a range like this because the precision was significantly lower throughout most of the tests. By definition the specificity and the false positive rate must sum to 1, which

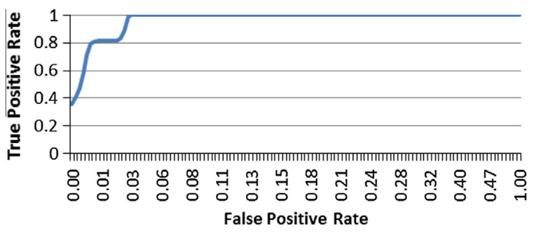

Fig. 5. ROC curve representing spam detection using the DenStream algorithm.

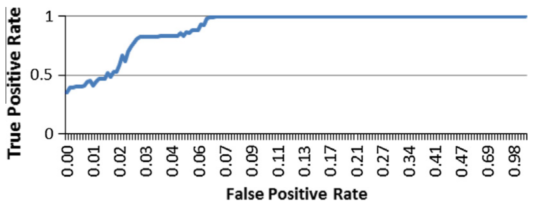

Fig. 6. ROC curve representing spam detection using the StreamKM++ algorithm.

is why when the specificity increases the false positive rate decreases accordingly. Because a low false positive rate is critical for spam detection, we wanted to explore its influence on the other metrics (Figs. 3 and 4). We found that DenStream was able to achieve between 0% and 2% false positive rate with all other metrics above 80%. StreamKM++ produced comparable results with a 3–10% false positive rate with the exception of precision and F-Measure.

3.5. Spam detection with StreamKM++ and DenStream combined

In our experimental setup of the combined approach, we tested three different values of ε to be used in the initial StreamKM++ step. The values of .5, 1.2 and 1.25 were chosen as they present a wide range of false positive rates with either perfect or near perfect recall. This means that all instances classified by StreamKM++ as normal were in fact normal Twitter users, and that all spam users were still considered as abnormal. The value of .5 was chosen as a test to see how well DenStream could remove 374 false positives. The range of epsilon values caused StreamKM++ to be inclusive of certain mistakes so that DenStream could refine the final predictions, particularly in F-Measure and Precision. Of interest is DenStream's ability to remove the majority of the false positives left by StreamKM++ while maintaining a high detection rate. This is evidenced by the test using the initial ε of .5 for StreamKM++, in which 374 false positives were reduced to 39 after DenStream. The *p* values for Specificity and False Positive Rate for StreamKM++ are high in this case because the parameters are not tuned to optimize these values; they are built to optimize Recall (Table 4). The combined approach had a recall value of 1 in Table 4 meaning that it detected 100% of the spammers. In the best case this approach produced an F-Measure of 88.5% and a Precision of 79.4% (Table 6). There is a slight difference in the numerical values reported in Tables 3, 4 and 6 because different parameters of the algorithms were used to optimize performance.

3.6. p-Values for spam detection

We employed the procedure outlined in Section 2.3 to evaluate the statistical significance of our spam detection. The *p*-values of our prediction are reported in Tables 3–6. In the majority of cases the *p*-value for any particular performance metric is 0, meaning that of the 1000 randomized predictions none outperformed our results.

3.7. ROC curves for spam detection

Because the false positive rate is so important to spam detection we include ROC curves for each algorithm. To generate these ROC curves, seen in Figs. 5 and 6, we re-ran both DenStream and StreamKM++ and varied the epsilon range. The goal was to make the range of values large enough so that we achieved as many False Positive Rates between 0 and 1 as possible. In both cases, if we used an epsilon value of 0, we would see that the False Positive Rates would be 1 as well as the Recall or True Positive Rate. We then continued to increase epsilon by an interval of .01 until we saw a False Positive Rate of 0. For DenStream, this occurred when epsilon was equal to 1.35 and for StreamKM++ it was 3.7. Once we had all of the Recall and False Positive Rates calculated, we removed the duplicate False Positive Rates taking the maximum associated Recall value and plotted them in Figs. 5 and 6.

4. Conclusion

Due to the increasing popularity and heavy use of social networks like Twitter, the number of spammers is rapidly growing. This has resulted in the development of several spam detection techniques [17–21]. This study has made three new contributions to the field of spam detection on Twitter. First we view spam identification as an anomaly detection problem. Secondly, we introduce 95 one-gram features from tweet text to the task of spam detection on Twitter. Finally, we use the stream of real-time tweets as well as user profile information with two stream-based clustering algorithms, DenStream and StreamKM++. When tested, these two approaches achieved 97.1% accuracy and 84.2% F-Measure and 94.0% accuracy and 74.8% F-Measure respectively. Our findings suggest the addition of one-gram features enhances spam detection. Although these algorithms independently demonstrated good detection, the combination of the two further improved all our metrics particularly recall and false positive rate to 100% and 2.2%, showing the value of the multi-layer approach to spam detection.

Acknowledgment

We would like to thank the Summer Research Institute at Houghton College for providing financial support for this study.

References

[1] M. Ackermann, C. Lammersen, M. Martens, C. Raupach, C. Sohler, K. Swierkot, StreamKM++: A Clustering Algorithm for Data Streams, SIAM ALENEX, 2010.
[2] M. Ankerst, M. Breunig, H. Kriegel, J. Sander, OPTICS: Ordering Points To Identify the Clustering Structure, SIGMOD, Philadelphia, 1999. pp. 49–60.
[3] D. Arthur, S. Vassilvitskii, k-Means++: the advantages of careful seeding, in: Proceedings of 18th Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027–1035.
[4] F. Benevenuto, G. Magno, T. Rodrigues, V. Almeida, Detecting Spammers on Twitter, Universidade Federal de Minas Gerais, 2010.
[5] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: massive online analysis, Journal of Machine Learning and Research 11 (2010) 1601–1604.
[6] C. Bohm, K. Kailing, H. Kriegel, P. Kroger, Density connected clustering with local subspace preferences, in: ICDMI, 2004, pp. 27–34.
[7] F. Cao, M. Ester, W. Qian, A. Zhou, Density-Based Clustering over an Evolving Data Stream with Noise, in: SIAM Conference Data Mining, Bethesda, 2006.
[8] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: International Conference on Knowledge Discovery in Databases and Data Mining (KDD-96), Portland, August 1996, pp. 226–231.
[9] Facebook's US User Growth Slows but Twitter Sees Double-Digit Gains, *EMarketer*(2012): n. pag. Web. 27 June 2012. <http://www.emarketer.com/Article.aspx?R=1008879>.
[10] S. Har-Peled, S. Mazumdar, Coresets for k-means and k-median Clustering and their Applications, in: Proc 36th ACM Sympos. Theory Comput, 2004, pp. 291–300.
[11] H. Kriegel, P. Kroger, I. Ntoutsi, A. Zimek, Towards subspace clustering on dynamic data: an incremental version of PreDeCon, in: Proc. First International Workshop on Novel Data Stream Pattern Mining Techniques, Washington DC, 2010, pp. 31–38.
[12] S. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory 28 (2) (1982) 129–136.
[13] M. McCord, M. Chuah, "Spam Detection on Twitter Using Traditional Classifiers", Autonomic and Trusted Computing (2011).
[14] A. Wang, Detecting spam bots in online social networking sites: a machine learning approach, in: 24th Annual IFIP WG 11.3 Working Conference on Data and Applications, Security, 2009.
[15] A. Wang, Don't Follow Me: Spam Detection in Twitter, in: Int'l Conference on Security and Cryptography (SECRYPT), 2009.
[16] I. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Elsevier, San Francisco, 2005.
[17] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, Guofei Gu, Analyzing spammers' social networks for fun and profit, in: Proceedings of the 21st International Conference on, World Wide Web, 2012, pp. 71–80.
[18] Zi Chu, Indra Widjaja, Haining Wang, Detecting social spam campaigns on twitter, in: Applied Cryptography and Network Security, Springer, Berlin Heidelberg, 2012, pp. 455–472.
[19] Chris Grier, Kurt Thomas, Vern Paxson, Michael Zhang, @ Spam: the underground on 140 characters or less, in: Proceedings of the 17th ACM Conference on Computer and Communications Security, ACM, 2010, pp. 27–37.
[20] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, Dawn Song, Design and evaluation of a real-time url spam filtering service, in: 2011 IEEE Symposium on Security and Privacy (SP), IEEE, 2011, pp. 447–462.
[21] Fabrıcio Benevenuto, Gabriel Magno, Tiago Rodrigues, Virgılio Almeida, Detecting spammers on twitter, in: Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS), 2010.