Assisted Labeling for Spam Account Detection on Twitter

Article

Accepted version

F. Concone, G. Lo Re, M. Morana, C. Ruocco

In Proceedings of the 2019 IEEE International Conference on Smart Computing (SMARTCOMP)

Publisher: IEEE

Assisted Labeling for Spam Account Detection on Twitter

Federico Concone, Giuseppe Lo Re, Marco Morana, and Claudio Ruocco
{federico.concone, giuseppe.lore, marco.morana, claudio.ruocco}@unipa.it
Università degli Studi di Palermo
Viale delle Scienze, ed.6, 90128, Palermo, Italy

***Abstract*—Online Social Networks (OSNs) have become increasingly popular both because of their ease of use and their availability through almost any smart device. Unfortunately, these characteristics make OSNs also target of users interested in performing malicious activities, such as spreading malware and performing phishing attacks. In this paper we address the problem of spam detection on Twitter providing a novel method to support the creation of large-scale annotated datasets. More specifically, URL inspection and tweet clustering are performed in order to detect some common behaviors of spammers and legitimate users. Finally, the manual annotation effort is further reduced by grouping similar users according to some characteristics. Experimental results show the effectiveness of the proposed approach.**

***Index Terms*—spam detection, social network, computer security**

I. Introduction

The popularity of Online Social Networks (OSNs) has rapidly grown up in the last few years, making social media part of everyone's life. Alongside with the most famous platforms, such as Facebook and Twitter, new OSNs were born providing different functionalities, such as instant messaging (e.g., WhatsApp and Telegram), or content sharing services, (e.g., Youtube and Instagram).

This popularity increase is mainly due to two factors: OSNs are very easy-to-use, and they are available ubiquitously, on every smartphone. Thus, people are encouraged to use OSNs very frequently to share personal information and communicate with other people they don't know in real life, overcoming communication barriers.

Thanks both to its APIs and developer policies, one of the most famous and investigated social network is Twitter. Moreover, *tweets* usually have high information content, being strictly related to popular events which involve many people in different parts of the world [8], [9].

Among the different research topics concerning Twitter analysis, spam accounts detection is one of the most investigated and relevant one. In general terms, the *spammers* are entities, real users or automated bots, whose aim is to repeatedly share messages that include unwanted content for commercial or offensive purposes [13], e.g., links to malicious sites that aim at spreading malware [17], phishing attacks, and other malicious activities.

In order to discourage these behaviors, social networks are continuously transforming and, together with them, spammers have also evolved adopting more sophisticated techniques that make it easy to evade security mechanisms [23].

The design and assessment of new spam detection techniques requires large datasets, whose collection and manual annotation are time consuming tasks. Moreover, being the lifetime of spammers very short, the datasets in the literature quickly become obsolete and almost useless.

In this paper, we present a system to support the annotation of large scale Twitter datasets by modeling the most relevant characteristics of spammers as compared to legitimate Twitter's users. In particular, after a first phase that deeply inspects URLs shared by users, a second step aims to find common patterns in users' timelines. Finally, similar users are grouped according to relevant features in order to reduce manual labeling efforts.

The remainder of the paper is organized as follows: Related work is outlined in Section II. The system for assisted labeling of spam account is described in Section III. Experimental results are presented in Section IV. Conclusions will follow in Section V.

II. Related Works

Spam detection on Twitter is one of the main topics in social media research and it has been investigated in many works. A comprehensive analysis of spam and compromised social network accounts is presented in [13].

Generally, a spam campaign is created by exploiting a number of *fake*, *compromised*, and *sibyl* accounts that operate in conjunction with social bots. For each of these threats, various detection techniques have been proposed [22]. The general idea is very simple and consists in attracting and deceiving possible attackers by means of an isolated and monitored environment. To this aim, some works propose the use of honeypots to analyze spamming activities. In [14], for instance, the authors present a social honeypot to collect spam profiles from social networking communities, such as MySpace and Twitter. Every time an attacker attempts to interact with the honeypot, an automated bot is used to retrieve some observable features, e.g., number of friends of the malicious users. Then, this set is analyzed to create a profile of spammers and train the corresponding classifiers.

A similar dynamic analysis is also performed in [19], where about 1.000 honeypots were used to accept *friend requests* and monitor friends activities in different OSNs (Facebook, MySpace and Twitter) for about 1 year. Results show that

four classes of spammers can be identified: *braggers*, posting malicious messages to their profiles; *displayers*, displaying spam content in their profiles; *posters*, sending messages to their victims through public posts on their walls; and *whisperers*, sending private messages to their victims.

Despite the advantages of performing a dynamic analysis on a controlled environment, the effort of creating a honeypot for each element to be analyzed is usually too high. For this reason, most works focus on static machine learning approaches capable of capturing some relevant features about the users and their interactions with their followers. In [7], three classifiers, i.e. Random Forest, Decision Tree, and Bayesian Networks, are used to learn nineteen features that reflect the spammers' behaviors.

In [2], authors describe *Twitter Sybils Detector*, a browser plug-in capable of classifying Twitter users according to a set of features. This system gets good results when asked to distinguish human from sybil, while the performances become lower when it is asked to deal with hybrid profiles.

This limitation is common to several works, suggesting that statistical features alone are not sufficient to correctly distinguish multiple classes of users. The reason is that spammers change their behavior over time to bypass security measures.

Since link sharing in one of the most common spam activities, many spam detection systems are based on URLs inspection [4].

Monarch [21], for instance, contains three different modules in order to capture URLs shared by web services, extract some features, and label every URL as spam or non-spam. Moreover, additional analyses are executed on other data, such as IP addresses and routing information.

The design of any spam detection technique requires two preliminary phases: collecting a great number of tweets, and labeling each element of the set as "spam" or "non-spam".

One of the first long-term data collection work is [15]. The dataset, captured by means of a honeypot, contains a total of 5.5 million tweets associated with both legitimate and malicious users.

HSpam14 [18] is a dataset of about 14 million tweets, collected by searching for trending topics and spammy words, i.e., words which are more likely to appear in spam tweets. According to the Twitter's policies, HSpam14 consists of a set of users' and tweets' IDs that should be used to fetch complete data through the Twitter APIs. Nevertheless, we observed that most of the requests fail because of different errors, e.g., *user account suspended*, *tweet ID not found*, and *account protected*, making HSpam14 quite unusable. This happens despite the dataset being only 3 years old, which highlights even more the early obsolescence of available collections.

The dataset provided in [3] consists of 600 millions public tweets, 6.5 millions of which are labeled as spam, and 6 millions as non-spam. The labeling has been performed according to the output of the Trend Micro's Web Reputation Service, that checks if an URL is malicious or not according to a reputation value given by users. Differently from HSpam14, this dataset contains the tweets and a fixed set of 12 features, but does not report the tweets' IDs that could be used to access other relevant information.

III. Twitter Dataset Labeling

The proposed *assisted labeling* approach is based on some intuitions that allow to distinguish between spammers and trustworthy users. Although all the strategies adopted by spammers to share unwanted contents are difficult to determine, academia and industry researchers agree on some behaviors and actions that are the basis of spamming.

One of the objectives of a spammer is to reach as many users as possible with a single message. For this reason, the first element that may probably identify a spam activity is the sharing of multiple messages containing same, or similar, information. This strategy is often complemented by connecting spam contents to a set of topics that are highly interesting to the user community, i.e., *trending topics*.

Another spamming strategy is the publication of malicious URLs that redirect legitimate users to sites containing malicious elements, e.g., malware and phishing sites [6]. Links are the most adopted way to disseminate malicious contents [7]; moreover, thanks to obfuscation strategies, spammer can easily hide the target URL in order to deceive the legitimate user.

Currently, both because of the tweets' character limit and the diffusion of URL blacklist services, a popular approach for spreading malicious links is the usage of URL shortening techniques. Twitter, for instance, provides an automatic service (*t.co*) that allows users to share long URLs in a tweet while maintaining the maximum number of characters allowed. However, since all shortened URLs look the same, the user is not aware of the actual destination address.

According to these characteristics, the labeling schema we propose is based on three phases: URL analysis, *similar tweets analysis*, and *similar users analysis*.

The whole process, summarized in Fig.1, aims at capturing the *modus operandi* of the users by analyzing their recent histories. Thus, each tweet in the timelines of the users (e.g., the latest 200 tweets) is analyzed. If enough malicious URLs are found within the timeline, then the corresponding user is labeled as *spammer*; otherwise, it is labeled as *genuine*. During the next phase, every timeline is checked in order to find similar tweets.

If results from URL and timeline analysis are consistent, i.e., both agree in considering the user as spammer or genuine, then the account is labeled consequently [5]. Otherwise, manual annotation is required. To minimize human effort in manual labeling, the remaining users are grouped together according to a set of features. The idea behind this final step is to create consistent groups of similar users, perform manual annotation of just a few samples per group, and then extend the label to the whole set.

A. URL Analysis

The simplest way to analyze a URL is by using blacklisting services, such as *Google Safe Browsing (GSB)*, that are able to alert whether a URL is malicious or not depending on the

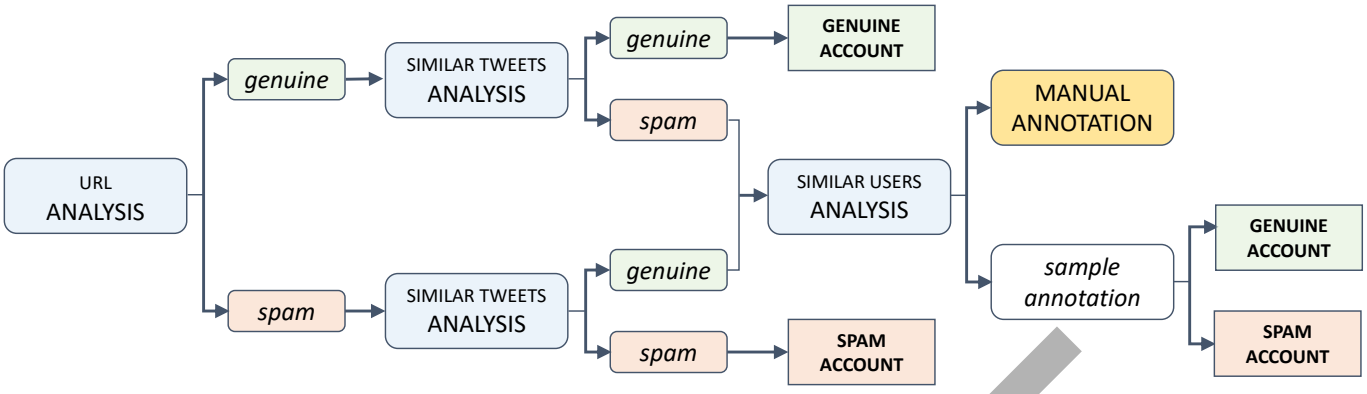

Fig. 1: The proposed labeling schema.

reports it has collected. Whereas such approach is the most common in the literature, its effectiveness is quite limited because blacklisting is not timely. In particular, some studies [11] highlighted that about four days are needed for a new website to be added to the blacklist, while most of the accesses on a tweet occur within two days from its publication. Even the URL shortening and safe-browsing services integrated with Twitter present similar limitations; for instance, it is not able to detect a malicious link that has been shortened twice (or more).

Moreover, blacklisting is able to capture *unsafe URLs* only; thus, a different kind of spammer, that continuously share *safe* links would never be detected.

As a consequence, our system not only relies on GSB as a tool to identify malicious URLs, but it also considers the amount of URLs within the users' timelines. To this aim, for each user, the total number of URLs, T, and the ratio R_{UT} between unique URLs, U, and T are computed. Preliminary experiments, carried out on a manually-labeled dataset, showed that good performances can be achieved by labeling as spammers those users who have shared at least one malicious URL, or whose spam index (R_{UT}) is less than 0.25.

B. Finding Similar Tweets

Another characteristic of spammers we want to capture is the sharing of similar contents over time. Thus, in order to correctly label those users that do not share malicious URLs, but keep posting same contents all over the timeline, the second phase consists in the identification of *near duplicates* tweets.

The core of the timeline analysis module exploits a clustering approach, known as *near duplicates clustering*, intended for grouping items, i.e., tweets, that are not identical copies but slightly differ from each other, e.g., by a few characters.

Correctly identifying near duplicates tweets requires the use of two different algorithms, namely MinHash and Locality-Sensitive Hashing (LSH) [10]. Nevertheless, before applying these algorithms, some tweet pre-processing is required.

In particular, every tweet is firstly represented as a set of tokens; then each token is analyzed in order to remove all those elements which do not contribute to the semantic, such as mentions, common symbols, punctuation marks and *stop-words*. Moreover, *URL expansion* and *stemming* are applied to obtain the actual URL and to group similar words respectively. As an example, supposing that *google.it* has been shortened through *bit.ly*, the following tweet:

```
@helloworld I'm writing this #tweet. Trying
      tokenization. bit.ly/1hxXbR7
```

would be transformed into:

```
write this tweet try token google.it.
```

The last step involves the choice of K, i.e., the number of consecutive elements to be considered as a single token. This choice deeply impacts on the system performances since the higher is K, the lower is the number of documents that will share the same word [16] and vice versa. For short documents, common values for K are in the range $[1, 3]$, while when dealing with longer texts, $K \geq 4$ is recommended. Since tweets are very short documents, we chose $K = 1$. Conversely, in HSpam14 [18] MinHash is applied in a slightly different way and other values of K are considered.

Having represented every tweet as a set of tokens, we need a distance function that allows to measure similarity between tweets. One of the most adopted metric is the Jaccard distance, which is computed as the ratio between the size of the intersection of the two documents and the size of their union. Such a distance, however, has a significant drawback since it can only be applied to two objects at a time. Thus, in order to create clusters of similar items, it would be required to analyze every possible pair of tweets, which is computationally expensive when dealing with large collections.

Another issue, which cannot be ignored, is that the number of tokens depends both on the amount of tweets to be analyzed and their size. This can deeply impact the memory required by the system. MinHash permits to overcome this limitation by providing a fast approximation of the Jaccard similarity using hash functions. In particular, the idea is to summarize the large sets of tokens into smaller groups, i.e., *signatures*, so that two tweets T_1 and T_2 can be considered similar if their signatures $Hash(T_1)$, and $Hash(T_2)$ are similar.

Input: Set of tokens S
N independent hash functions
Output: $< H_m(1), H_m(2), \ldots, H_m(N) >$
for $i = 1 : N$ **do**
  $H_m(i) \leftarrow \infty$;
end
forall *token* $\in S$ **do**
  for $i = 1 : N$ **do**
    if $Hash_i(token) < H_m(i)$ **then**
      $H_m(i) \leftarrow Hash_i(token)$;
    end
  end
end

Algorithm 1: MinHash signature.

Algorithm 1 describes the MinHash signature generation when using N hash functions. For every hash function h_i and for every token t_j a value is computed as $h_i(t_j)$. Then, the $i-th$ element of the signature is:

$$s_i = \min_j \ h_i(t_j). \tag{1}$$

Using Minhash we can solve the problem of comparing large datasets by compressing every tweet into a signature. However, we still need to perform pairwise comparisons in efficient way. LSH - *Locality-Sensitive Hashing* - is the algorithm which best resolves this problem by using a hash table to group similar elements into buckets. The hash functions it uses are purposely made to maximize the probability of similar tweets to be hashed into the same bucket.

Essentially, LSH groups all the MinHash signatures into a matrix, then splits it into B bands, each composed of R rows, and a hash value for each band is computed. If two tweets fall into the same bucket for at least one band, then they are considered as potential near-duplicates and they can be further inspected through real or approximate Jaccard similarity.

The combination of MinHash and LSH allows to group the tweets contained in the users' timelines into set of clusters. Then, a user can be labeled as spammer or genuine according to the characteristics of these clusters. In Section IV, we present the experiments carried out in order to identify and validate the set of features that best discriminate between spammers and genuine users.

C. Finding Similar Users

The very last phase of the proposed annotation schema involves the use of another clustering algorithm, namely *Quality Threshold Clustering* (QTC) [12], which is aimed at reducing the manual annotation effort when the previous stages provide incongruous results. Thus, Quality Threshold Clustering is applied only on the set of users that were not labeled as *genuine* or *spam* during the previous phases. The idea behind this phase is to group *very similar* users so as to perform manual annotation of just a few samples per group, and then extend the label to the whole set.

Input: Set of points P, MinClusterSize m,
MaxClusterDiam d
Output: Set of generated clusters S_c
$S_c \leftarrow \emptyset$
if $|P| \leq 1$ **then**
  $S_c \leftarrow P$
else
  while $|P| \geq 1$ **do**
    foreach $i \in P$ **do**
      $flag \leftarrow true$
      $C_i \leftarrow \{i\}$
      while $(flag = true) \wedge (C_i \neq P)$ **do**
        foreach $j \in (P - C_i)$ **do**
          if $distance(i, j) > \frac{d}{2}$ **then**
            $flag \leftarrow false$
          else
            $C_i \leftarrow C_i \cup \{j\}$
          end
        end
      end
    end
    $S_A \leftarrow \{C_1, C_2, \ldots C_{|P|}\}$
    $k \leftarrow \arg\max_{C_i \in S_A} |C_i|$
    if $|C_k| \geq m$ **then**
      $S_c \leftarrow S_c \cup \{C_k\}$
      $P \leftarrow P - C_k$
    else
      foreach $i \in P$ **do**
        $S_c \leftarrow S_c \cup \{\{i\}\}$
        $P \leftarrow P - \{i\}$
      end
    end
  end
end

Algorithm 2: Quality Threshold Clustering (QTC).

Unlike other clustering techniques, QTC does not require the a priori specification of the number of clusters to be found. Instead, elements are progressively grouped while maintaining the *quality* of each cluster above a certain threshold. To this aim, two parameters must be defined, i.e., the maximum cluster diameter (d), and the minimum number of elements (m) a cluster has to contain. QTC works as follows: for every iteration, for every data point, it finds the closest candidate to be added (the one which causes the minimum increase in diameter) to the cluster. This is repeated until no points can be added without exceeding the value of d. At the end of each iteration, if the largest cluster has more elements than m, then it is *closed* and its points are not considered in the following steps. Conversely, if the largest cluster's size is lower than m, the algorithm ends and the remaining points are considered as clusters containing a single element.

In its early implementation, QTC is computationally intensive and time consuming, which is the reason behind the great presence of efficient variations of the algorithm. In this work we adopted an improved version of QTC (see Algorithm 2)

that considers as candidates only those points within half of the maximum distance from the centre.

Since the output clusters differ in size and quality, we only considered the first cluster C_{best}, which is the biggest and most reliable one. Thus, only the users in C_{best} are labeled, while the remaining users will be evaluated through the manual annotation procedure.

IV. Experimental Analysis

Experimental analysis has been focused on the tuning and evaluation of the algorithms discussed so far.

The first set of experiments aimed at finding the best set of parameters for the *similar tweets analysis* module, i.e., the quadruple (N, K, B, J) for MinHash and LSH, where N is the number of hash functions, K is the number of consecutive tokens, B is the number of bands, and J is the minimum Jaccard distance to consider two tweets similar. Whereas N has been set to 200 as suggested in the literature, the other parameters have been selected varying their values as following: $K = \{1, 2, 3\}$, $B = \{5, 10, 20, 40, 50\}$, and $J = \{0.5, 0.6, 0.7, 0.8\}$. Values for K and J have been chosen amongst the most used in the literature, whereas those for B have been selected according to [18].

In order to evaluate the results achieved by each quadruple, a reference dataset was used. In particular, we exploited the dataset in [20], which is composed by pairs of tweets manually labeled with a similarity score that varies from 1 (dissimilar) to 5 (equal). A pairwise similarity criterion was used to transform these labels into a ground-truth about *clusters* of tweets. For instance, if a tweet t_1 is considered similar to t_2, and t_2 is also similar to t_3, then t_1 and t_3 are similar and the three tweets belong to the same cluster. Furthermore, to ensure a high degree of similarity among tweets belonging to the same cluster, we considered only those pairs whose similarity score is at least 3, i.e., those labeled by [20] as "strong near duplicates".

The performance of MinHash and LSH were compared in terms of f-score, as reported in Table I. According to these experiments, the best values are $K = 1$, $B = 50$, and $J = 0.5$, which allow to achieve a f-score of 0.69.

Once the parameters have been set, the next experiments were intended to select the most suitable set of features for distinguishing spammers from genuine users. For instance, given a timeline of P tweets, it is reasonable to assume that tweets shared by a *genuine* user would be very different from each other, while *spammers* are likely to share similar contents. Thus, MinHash and LSH would produce about P clusters on the timeline of a *genuine* user, and Q clusters, with $Q \ll P$, in the case of a *spammer*.

In order to obtain a more accurate representation of *spam* and *genuine* classes, the output of MinHash and LSH needs to be further summarized. To this aim, different features were considered, such as the size of the largest cluster (f_1), mean size of clusters (f_2), number of clustered tweets (f_3), size of the smallest cluster (f_4), and number of generated cluster (f_5).

		J			
K	***B***	***0.5***	***0.6***	***0.7***	***0.8***
1	***5***	0.249	0.248	0.248	0.247
	10	0.297	0.296	0.297	0.290
	20	0.477	0.467	0.436	0.371
	40	0.675	0.587	0.468	0.376
	50	**0.688**	0.589	0.468	0.376
2	***5***	0.242	0.242	0.242	0.242
	10	0.250	0.250	0.250	0.249
	20	0.287	0.283	0.271	0.260
	40	0.389	0.328	0.276	0.261
	50	0.403	0.330	0.276	0.261
3	***5***	0.241	0.241	0.241	0.241
	10	0.247	0.247	0.247	0.246
	20	0.265	0.263	0.257	0.253
	40	0.310	0.285	0.259	0.254
	50	0.315	0.285	0.259	0.254

TABLE I: F-score obtained by calibration phase for MinHash and LSH algorithms.

ratio	***S1***	***S2***	***S3***	***S4***	***S5***	***S6***	***S7***
0.3	0.790	0.793	**0.794**	**0.794**	0.787	**0.794**	0.791
	0.768	0.772	0.772	**0.773**	0.765	**0.773**	0.769
0.4	0.784	0.792	**0.803**	0.788	0.782	0.799	0.791
	0.760	0.771	**0.784**	0.764	0.759	0.778	0.767
0.5	0.787	0.797	**0.800**	0.785	0.779	0.799	0.790
	0.763	0.777	**0.781**	0.761	0.755	0.779	0.765
0.6	0.790	0.796	**0.807**	0.786	0.778	0.800	0.793
	0.767	0.776	**0.789**	0.762	0.755	0.778	0.768
0.7	0.780	0.794	**0.811**	0.779	0.770	0.797	0.789
	0.752	0.774	**0.792**	0.757	0.746	0.775	0.762
0.8	0.786	0.794	**0.812**	0.785	0.775	0.793	0.788
	0.760	0.774	**0.794**	0.764	0.752	0.769	0.760

TABLE II: Accuracy (gray rows) and f-score achieved while varying the ratio of *spammers* and *genuine* users in the range [0.3,0.8]. For each experiment, the following set of features were considered: $S_1 = \{f_1, f_2\}$, $S_2 = \{f_1, f_2, f_3\}$, $S_3 = \{f_1, f_2, f_5\}$, $S_4 = \{f_1, f_2, f_4, f_5\}$, $S_5 = \{f_1, f_2, f_3, f_5\}$, $S_6 = \{f_1, f_2, f_3, f_4\}$, and $S_7 = \{f_1, f_2, f_3, f_4, f_5\}$.

Then, the effectiveness of different subsets of features was evaluated. For these experiments we relied on a subset of the data in HSpam14 [18], which contains 14 million labeled tweets. However, since our aim is to label users, we sampled some of the tweets in HSpam14, retrieved information about the authors, and then labeled them according to the original tweet's label. Tests were run while varying the ratio between genuine users and spammers (e.g., a ratio of 0.3 indicates that the dataset is composed of 30% spammers and 70% genuine users), and using 7 subsets of features (see Table II). Although all the subsets S_i provide similar performances, we have selected the set S_3 because of its compactness. In particular, by considering the features in S_3, i.e., the *maximum* size and *average* size of clusters (f_1, f_2), and the *number* of clusters (f_5), the system achieved an accuracy of 0.79 and a f-score of 0.77.

The last set of experiments focused on the assessment of QTC. In this case, we would need to define the values of the minimum cluster size (m), and the maximum distance (d) between elements that belong to the same cluster. However, given that we want to analyze only users belonging to the

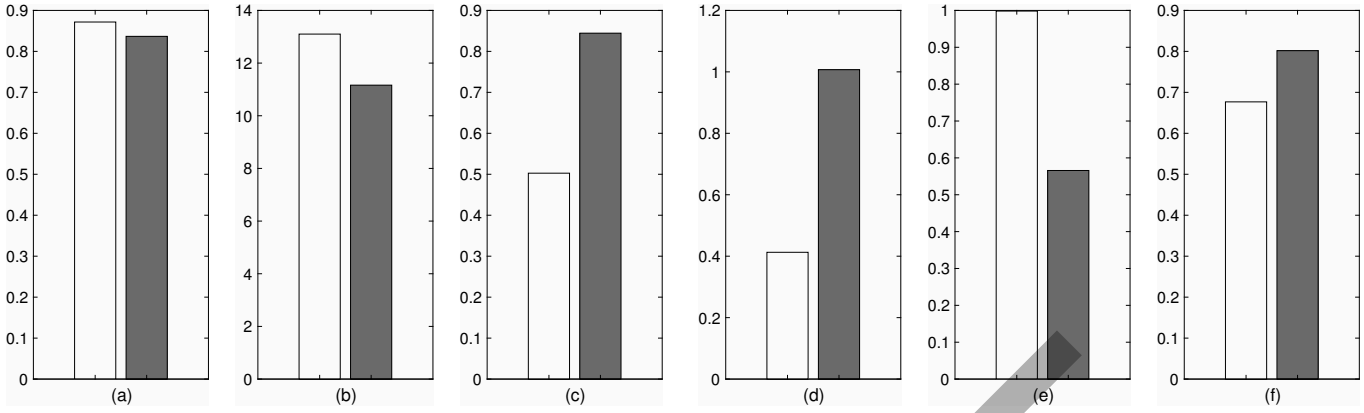

Fig. 2: Features computed on spammers (black bars) and genuine users (white bars): timeline fullness (a), corresponding to the number of tweets divided by 200; average number of tweets per day (b); average number of URLs (c), *hashtags* (d), *mentions* (e), and *spamwords* (f) per tweet.

biggest cluster computed by QTC, the value of m can be neglected.

Since this phase aims at comparing users according to their timelines, we identified characteristics that would capture significant differences between spammers and genuines. In particular, we considered as candidate features the *timeline fullness* (number of tweets in a timeline, divided by 200), the daily *frequency* of *tweets*, and the average number of URLs, *hashtags*, *mentions*, and *spamwords* per tweet. Fig. 2 shows the average values of such features computed for spammers (black bars) and genuine users (white bars) on a training set of labeled data. According to these results, we selected URLs, *hashtags*, and *mentions* frequencies.

The best value of d was found by evaluating the performance of QTC in terms of the *number of elements* within the first cluster, and the *error* computed as the ratio of the number of elements of minority class within the cluster and the cluster size.

Results obtained varying d in the range $[0.4, 2.0]$ are summarized in Fig. 3 and Fig. 4. For each experiment, different amounts of spammer and genuine users were considered. By focusing on the ratio $10/90\%$, that is close to the actual distribution of spammers on Twitter, the two plots suggest that a good trade-off between large clusters, which are desired to reduce the manual annotation phase, and low error is obtained with $d = 1$. As shown in Fig. 4, this value allows to label about the 45% of users with a margin of error of 4%.

Finally, in order to assess the overall performances of the automatic labeling procedure, a dataset was collected using the Twitter APIs. Tweet collection was performed at regular intervals by exploiting a set of keywords that include both trending topics and "spammy" words, such as money gain and adult contents [18]. For each tweet, the author and the list of followers have been extracted, together with standard tweet-related data, such as the tweet identifier, creation date, and so on. Extending the search to the followers of potential spammers allowed us to increase the probability of finding spammers. The complete list of authors and followers has then been processed to obtain also the latest 200 tweets contained in each timeline. As a result of this procedure we collected almost 8 million tweets and 40 thousands users.

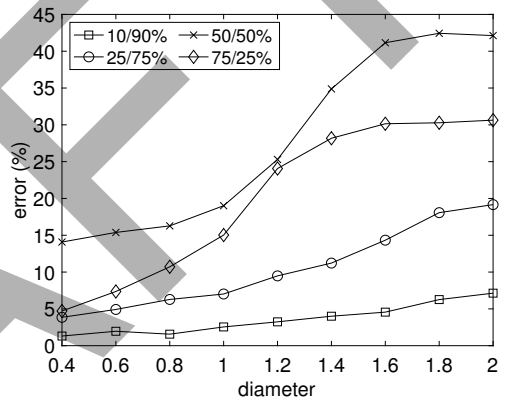

Fig. 3: Error within the first cluster while varying the parameter d.

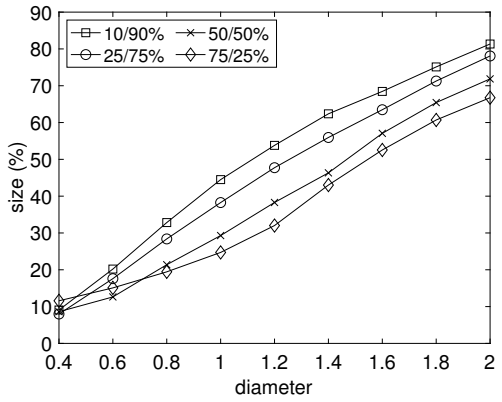

Fig. 4: Percentage of elements grouped in the first cluster while varying the parameter d.

The dataset has been analyzed by applying the proposed procedure and the outcomes of the labeling process are shown

Number of users collected	40.823
Automatically labeled as genuine	20.007
Automatically labeled as spammers	2.190
Grouped in the first QT cluster	6.780
Needing further manual inspection	11.846
Number of tweets collected	8.010.147
Containing URLs	2.330.558
Containing hashtags	1.640.521
Containing user mentions	4.334.056

TABLE III: Output of the detection/labeling process on the dataset collected.

in Table III. In particular, the *URL analysis* and *timeline analysis* steps allowed to automatically detect 20.000 legitimate users and about 2.000 spammers. These results were compared with a ground-truth obtained by manually labeling the users we collected, showing an average accuracy of about 80%. In particular, we measured that the accuracy of the assisted labeling system reaches the maximum value of 95% when detecting true genuine users, while this percentage is lower when dealing with spammers (about 70%). These values are not surprising and reflect the fact that activities carried out by genuine users are quite predictable, while spammers frequently vary their *modus operandi* in order to elude spam detection systems.

The users that were not automatically labeled after the application of MinHash and LSH have been analyzed by means of QT clustering. The biggest cluster produced by QT, consisting of almost 7.000 users, i.e., about 35% of the remaining users, was confidently labeled as genuine. All the other users, about 11.000, would need to be further manually inspected to have a label assigned. Thus, about 75% of the users can be labeled in a semi-automatic way, leading to a significant reduction in manual efforts needed to inspect the whole dataset.

V. Conclusion

Since the design of novel spammer detection methods requires a first phase of data collection and data labeling, in this paper we presented a system to support the annotation of large-scale Twitter datasets by modeling the most common behaviors of Twitter spammers, i.e. URLs sharing and presence of patterns in tweets.

Although malicious URLs can be detected by relying on third-party blacklisting services, we noticed that these systems alone are not sufficient to detect any form of link-based spam contents. Thus, a URL analyzer taking into account a greater number of features has been described. As regards the analysis of recurring topics and near-duplicate contents, a combination of MinHash and Local-Sensitive Hashing algorithms has been presented.

Such a system aims to provide researchers with a tool to speed-up the automatic annotation of large-scale datasets. In particular when the presence of malicious URLs or frequent patterns is not so evident, the proposed labeling approach exploits a Quality Threshold clustering algorithm to group user with similar characteristics in order to drastically reduce the manual annotation efforts.

At first, experimental evaluation has been focused on the tuning of parameters for the adopted techniques, and on selecting the best set of features which permits to distinguish between spammer and genuine users.

Moreover, in order to validate the effectiveness of the proposed method, a dataset was collected and analyzed. Results showed that almost 75% of the accounts contained in the dataset can be labeled by means of the technique we propose with an average accuracy of about 80%.

As future works we plan to extend our analysis to non-english tweets, using different pre-processing algorithms and identifying both spam-words and stop-words for every language. Moreover, a continuous analysis of the user's behavior could allow to estimate the *reputation* of each user. These values, similarly to those assigned to agents cooperating in distributed systems [1], could represent another feature to be analyzed by the spam/genuine classification algorithm.

Finally, given the Twitter's privacy policies, we are not able to release the dataset we collected, but we want to release a public version of our labeling system. Given the huge amount of data to be processed, the software will exploit different threads to perform parallel analysis on multiple data at the same time.

References

[1] Vincenzo Agate, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. *A Simulation Framework for Evaluating Distributed Reputation Management Systems*, pages 247–254. Springer International Publishing, Cham, 2016.

[2] M. Alsaleh, A. Alarifi, A. M. Al-Salman, M. Alfayez, and A. Almuhaysin. Tsd: Detecting sybil accounts in twitter. In *2014 13th International Conference on Machine Learning and Applications*, pages 463–469, Dec 2014.

[3] 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, June 2015.

[4] Chao Chen, Sheng Wen, Jun Zhang, Yang Xiang, Jonathan Oliver, Abdulhameed Alelaiwi, and Mohammad Mehedi Hassan. Investigating the deceptive information in twitter spam. *Future Gener. Comput. Syst.*, 72(C):319–326, July 2017.

[5] F. Concone, G. Lo Re, M. Morana, and C. Ruocco. Twitter spam account detection by effective labeling. In *Proceedings of the Third Italian Conference on Cyber Security*, volume 2315, 2019.

[6] Federico Concone, Alessandra De Paola, Giuseppe Lo Re, and Marco Morana. Twitter analysis for real-time malware discovery. In *2017 AEIT International Annual Conference (2017 AEIT)*, Cagliari, Italy, sep 2017.

[7] M. Fazil and M. Abulaish. A hybrid approach for detecting automated spammers in twitter. *IEEE Transactions on Information Forensics and Security*, pages 1–1, 2018.

[8] S. Gaglio, G. Lo Re, and M. Morana. Real-time detection of twitter social events from the user's perspective. In *IEEE International Conference on Communications (ICC2015), 2015*, pages 2810–2815, 2015.

[9] Salvatore Gaglio, Giuseppe Lo Re, and Marco Morana. A framework for real-time twitter data analysis. *Computer Communications*, 73, Part B:236 – 242, 2016. Online Social Networks.

[10] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[11] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @ spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 27–37. ACM, 2010.

[12] L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring Expression Data: Identification and Analysis of Coexpressed Genes. *Genome Res.*, 9(11):1106–1115, 1999.

[13] Ravneet Kaur, Sarbjeet Singh, and Harish Kumar. Rise of spam and compromised accounts in online social networks: A state-of-the-art review of different combating approaches. *Journal of Network and Computer Applications*, 112:53 – 88, 2018.

[14] Kyumin Lee, James Caverlee, and Steve Webb. The social honeypot project: Protecting online communities from spammers. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1139–1140, New York, NY, USA, 2010. ACM.

[15] Kyumin Lee, Brian David Eoff, and James Caverlee. Seven months with the devils: A long-term study of content polluters on twitter. In *ICWSM*, pages 185–192, 2011.

[16] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[17] A. De Paola, S. Gaglio, G. Lo Re, and M. Morana. A hybrid system for malware detection on big data. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 45–50, April 2018.

[18] Surendra Sedhai and Aixin Sun. Hspam14: A collection of 14 million tweets for hashtag-oriented spam research. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 223–232, New York, NY, USA, 2015. ACM.

[19] 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.

[20] Ke Tao, Fabian Abel, Claudia Hauff, Geert-Jan Houben, and Ujwal Gadiraju. Groundhog day: near-duplicate detection on twitter. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1273–1284. ACM, 2013.

[21] 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.

[22] Tingmin Wu, Sheng Wen, Yang Xiang, and Wanlei Zhou. Twitter spam detection: Survey of new approaches and comparative study. *Computers & Security*, 2017.

[23] C. Yang, R. Harkreader, and G. Gu. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8):1280–1293, Aug 2013.