Twitter Spam Account Detection by Effective Labeling

Federico Concone, Giuseppe Lo Re, Marco Morana, and Claudio Ruocco

University of Palermo, Viale delle Scienze, ed. 6 - 90128 Palermo, Italy
{firstname.lastname}@unipa.it

Abstract. In the last years, the widespread diffusion of Online Social Networks (OSNs) has enabled new forms of communications that make it easier for people to interact remotely. Unfortunately, one of the first consequences of such a popularity is the increasing number of malicious users who sign-up and use OSNs for non-legit activities. In this paper we focus on spam detection, and present some preliminary results of a system that aims at speeding up the creation of a large-scale annotated dataset for spam account detection on Twitter. To this aim, two different algorithms capable of capturing the spammer behaviors, i.e., to share malicious *urls* and recurrent contents, are exploited. Experimental results on a dataset of about 40.000 users show the effectiveness of the proposed approach.

Keywords: Social Network Security · Spam Detection · Twitter Data Analysis.

1 Introduction

Online Social Networks (OSNs) are platforms through which a multitude of people can interact remotely. Nowadays different types of OSNs are available, each with its own characteristics and functionalities depending on the purpose and target for which it is intended. The simplicity of use of these tools, together with the diffusion of smart personal devices that allow continuous access to the network, stimulate users to overcome some communication barriers typical of real life. As a result, people are encouraged to share personal information, even with entities (people or other systems) that are actually unknown.

Although the number of OSNs is ever increasing, many researches have focused on Twitter analysis because the information content of the *tweets* is usually very high, being strictly related to popular events which involve many people in different parts of the world [9, 10]. Moreover, it is extremely easy to access the Twitter stream thanks to the API platform that provides broad access to public data that users have chosen to share.

Among the different analyses concerning Twitter, spam accounts detection is one of the most investigated and relevant one. In general terms, *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, in order to spread malwares, phishing attacks, and other harmful activity [5].

Spam detection is part of the unending fight between cops and robbers. In order to discourage malicious behaviors, social networks are continuously transforming and, as a consequence, spammers have also evolved, adopting more sophisticated techniques that make it easy to evade security mechanisms [23]. Since the design of new spam detection techniques requires stable and annotated datasets to assess their performances, such a dynamism makes the datasets in the literature quickly obsolete and almost useless. Moreover, providing the ground-truth of a huge amount of data is a time consuming task that, in most cases, is still performed manually.

In this paper, we present the preliminary results of a work that aims at modeling Twitter spammers' behavior in order to speed up the creation of large-scale annotated datasets.

The system consists of different software modules whose purpose is to capture certain aspects of the spammers' *modus operandi*. In particular, we focused on two common characteristics, namely sharing malicious *urls*, and the presence of messages with the same information content.

The remainder of the paper is organized as follows: Related work is outlined in Section 2. The spammer detecting system is described in Section 3. Experimental results are presented in Section 4. Conclusions will follow in Section 5.

2 Related Work

In recent years, the spam detection on Twitter has been investigated in many works.

The different ways in which malicious users operate can be categorized according to the method they adopt to disseminate illegitimate information [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 [21]. 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 able to collect spam profiles from social networking communities. 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.

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 [7]. For this reason, most works focus on static machine learning approaches capable of capturing some relevant features about the users and their interactions. In [8], three classifiers, i.e. Random Forest, Decision Tree,

and Bayesian Networks, are used for learning nineteen features that reflect the spammers' behaviors.

Another work using machine learning approach to identify malicious accounts is presented in [2]. The authors developed a browser plug-in, called TSD (Twitter Sybils Detector), capable of classifying a Twitter profile as *human*, *sybil*, or *hybrid* according to a set of seventeen features. Such a system provides good results when distinguishing human from sybil, but the performances get worse when dealing 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.

A strategy that is becoming increasingly popular is to use *urls* as a key element to recognize a spammer [4]. A system exploiting *urls* to detect spammers within social networks is Monarch [20]. Here, three different modules aim to capture *urls* submitted by web services, extract a feature set (e.g., domains tokens, path tokens, path length), and label a specific *url* as spam or non-spam. In addition, supplementary data, such as IP addresses and routing information, are collected using DNS resolver and IP analysis.

All these techniques require two preliminary phases: collecting a great number of tweets, and label 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 probably the most diffused dataset for spam detection on Twitter. This dataset contains the IDs of 14 million tweets obtained by searching for some trending topics. These identifiers should be used to access the original tweets through the standard Twitter APIs. Unfortunately, although it has been released just a few years ago, we observed that most of the requests fail because of different errors, i.e., *user account suspended*, *tweet ID not found*, and *account protected*.

Conversely, the dataset described in [3] consists of over 600 million public tweets, 6.5 million of which are labeled as spam and 6 million as non-spam. The labeling is performed according to the output of the Trend Micro's Web Reputation Service, that checks if an *url* is malicious or not. If so, they label the corresponding tweet as Twitter spam. Differently from HSpam14, this dataset contains the tweets and a fixed set of 12 features, but does not report the tweet IDs that could be used to access other relevant information.

3 Twitter Dataset Labeling

In this section we present a novel approach that aims at supporting the labeling of large-scale Twitter datasets.

The design of a *smart labeling* technique requires the definition of some criteria that allow to distinguish between spammers and trustworthy users [1,17]. The official Twitter documentation defines the spam activity as a series of behaviors

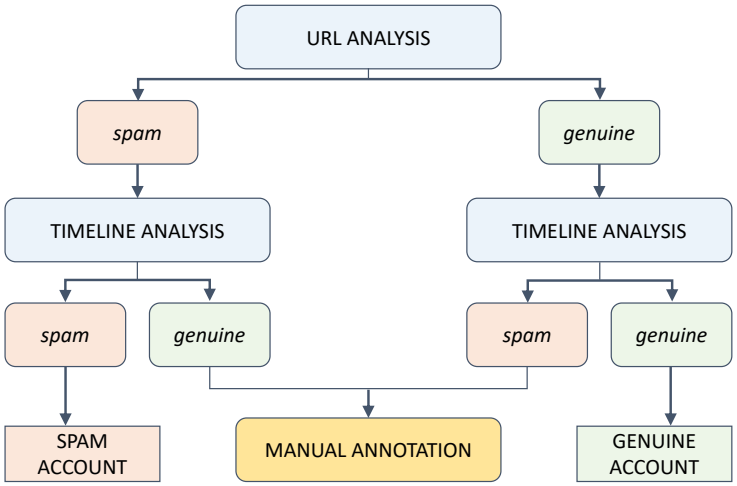

Fig. 1. Overview of the proposed automatic labeling schema.

and actions that negatively affect other users and violate the rules of the social network. Considering that malicious behaviors are constantly evolving, it is not possible to provide a definitive set of them, but we can identify some strategies that are common to most of the spammers.

The first point to consider is the publication of malicious *urls* that direct to phishing sites or induce users to download unwanted software [6]. Detecting such links is not simple because spammers adopt strategies that obfuscate the target *url*, thus deceiving the end user. For this reason, despite the possible countermeasures, links are the easiest way to disseminate malicious contents [8].

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 services. 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 respected. However, since all shortened *urls* look the same, users may not be aware of the actual destination address.

Another typical spammers' behavior is to repeatedly publish duplicate messages, or messages with the same information content. This strategy is often complemented by exploiting a set of topics that are highly interesting to the user community. Generally, OSNs allow legitimate users to report suspicious behaviors in order to let the administrators verify whether a given account is malicious or not. However, manually detecting this type of behavior is time-consuming and resource-intensive.

On the basis of these characteristics, the labeling schema we propose is based on two phases: *url* analysis, and similar tweets discovery.

As summarized in Fig.1, a tweet is firstly analyzed to verify whether it contains links or not. Either way, the result of this check provides only a preliminary

outcome that needs to be further investigated. Thus, the next phase consists of a timeline (e.g., the last 200 tweets) analysis for every user. If both results are consistent, i.e., both agree in considering the user as spammer or genuine, then the account is labeled consequently. Otherwise, the automatic labeling fails and a manual annotation step is required.

3.1 URL Analysis

Not surprisingly, tweets containing links are more likely to get re-tweeted, which is the primary goal of most spammers. For this reason, the presence of a *url* in a tweet is frequently indicative of potential spam activities.

Most of the works in the literature perform link analysis by exploiting blacklist services, e.g., *Google Safe Browsing* (GSB), that are able to find whether a given *url* is malicious or not. Unfortunately, the effectiveness of such a solution is quite limited because these services usually take an average of four days to add a new website to the blacklist, while most of the accesses on a tweet occur within two days from the publication [12]. Even the *url* shortening and safe-browsing services integrated with Twitter present some limitations. This system, for instance, is not able to detect a malicious link that have been shortened twice or more.

Another point to be considered is that by relying on these tools only, a user continuously sharing the same *safe* link, or the same kind of content, although being a spammer, would never be recognized.

For these reasons, the *url* analyzer we propose takes into account a greater number of features related to link sharing activity. In particular, three factors are considered while analyzing a tweet: i) the presence of malicious *urls* according to GSB, ii) the total number of *urls*, T, and iii) the ratio R_{UT} between the number of unique *urls*, U, and T. The value of T permits also to discard those users that have not published a sufficient number of *urls* in their timelines.

Preliminary experiments showed that accounts satisfying one of the two following conditions can be labeled as spammers for this module: i) at least one malicious *url* is found by GSB; ii) the ratio $R_{UT} \leq 0.25$ and $T \geq 50$. Otherwise, the account is considered genuine.

3.2 Finding Similar Tweets

In many applications, it is often necessary to divide data into homogeneous groups, *clusters*, whose elements share the same characteristics. Several clustering techniques have been proposed in the literature [22].

The second phase of our annotation schema is based on a clustering approach, known as *near duplicates clustering*, intended for grouping items, i.e., tweets, that are identical copies or slightly differ from each other, e.g., by a few characters.

The aim of this phase is to measure the degree of similarity between the tweets contained in the timeline of each user. Near-duplicated tweets can be found by using MinHash and Locality-Sensitive Hashing (LSH) [11] alghoritms.

Table 1. Tweet pre-processing.

Remove all non-english tweets	Because of the language-dependency of some tokenization algorithm, e.g., stemming, only english tweets have been maintained.
Remove mentions	Mentions are not semantically significant as they only allow users to redirect their tweets to specific users.
Convert text to lower case	There are no semantic differences between words written in lowercase or uppercase.
Apply stemming	Group words having the same stem (root).
Remove # and common symbols	The character #, as well as punctuation marks, are frequently used and can negatively affect near-duplicates detection.
Expansion of urls	Follow all the re-directions of the *urls* included in the tweets.
Remove stop-words	Stop-words, such as conjunctions, articles and prepositions, can be omitted without altering the meaning of the tweet.
Normalizing accented characters	Conversion of accented letters into the corresponding non-accented versions.

Nevertheless, a few steps need to be performed before the applications of these two algorithms.

The first step aims to represent tweets as sets of tokens. These can be defined either as consecutive characters, called *shingles*, or as single words composing the document. The latter is the one we used.

The second step includes different pre-processing operations, summarized in Table 1, that are needed in order to improve the performances of MinHash and LSH, as suggested in [18]. According to their model, we chose to remove all those elements which do not contribute to the semantic of the tweet, such as punctuation marks and *stop-words*. Moreover, we added some more steps, such as *url expansion* and *stemming*. For instance, the tweet:

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

would be transformed into:

```
write 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. A good rule is to set K equal to 1, 2, or 3 for small to medium sized documents, whilst 4 or 5 are reasonable values for very large documents. Since the tweets are very short documents, we chose $K = 1$, while in [18] authors used all the aforementioned values.

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.

Representing every document as a set of tokens makes it easier to compute the similarity between sets of documents. A simple similarity metric is the Jaccard distance, which is the ratio between the size of the intersection of the two documents and the size of their union. Since the Jaccard similarity can only be applied to two objects at a time, it is required to analyze every possible pair of documents in order to create clusters of similar items. When dealing with a high number of document, this process is computationally expensive, being the number of comparisons given by the binomial coefficient:

$$\binom{N}{K} = \binom{N}{2} = \frac{N!}{2!(N-2)!} = \frac{N(N-1)}{2} \approx \frac{N^2}{2}. \tag{1}$$

Furthermore, a second factor that cannot be ignored is that the number of tokens depends both on the amount of documents to be analyzed and their size. To overcome such a limitation, the MinHash algorithm permits to approximate the Jaccard similarity by using hash functions. The idea is to summarize the large sets of tokens into smaller groups, i.e., *signatures*, so that two documents D_1 and D_2 can be consider similar if their signatures $Hash(D_1)$, and $Hash(D_2)$ are similar.

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:

$$s_i = \min_j \ h_i(t_j). \tag{2}$$

Although MinHash solves the problem of comparing large datasets by compressing every document into a signature, we still need to perform pairwise comparisons in efficient way. This is the reason behind the usage of LSH - *Locality-Sensitive Hashing* - which exploits a hash table and maximizes the probability of similar documents 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. Then, a hash value for every document, for every band, is computed. If two documents 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.

By applying MinHash and LSH, the tweets contained in the users' timelines are grouped into sets of clusters. The process of labeling a user as spammer or genuine depends on the characteristics of these clusters. To this aim, different features describing the size and the number of clusters were considered. In the next Section, experimental results achieved while varying the feature set will be presented.

4 Experimental Analysis

The first set of experiments aimed at finding the best set of parameters for MinHash and LSH, i.e., the quadruple (N, K, B, J), 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 $J = \{0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8\}$.

In order to evaluate the results achieved by each quadruple, a reference dataset was used. In particular, we exploited the dataset in [19], 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.,"strong near duplicates".

The performance of MinHash and LSH were evaluated in terms of precision, recall, and f-score. Fig. 2 shows the f-score obtained for each triple (K, B, J). 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 in order to distinguish spammers from genuine users. For instance, assuming that the timeline of a user is composed of N tweets, we can expect that for a *genuine* user the number of clusters is close to N; whilst for a *spammer* this number would be M, with $M \ll N$.

Thus, in order to obtain a compact representation of *spammer* and *genuine* classes, feature vectors have been created by considering *mean*, *variance*, and *standard deviation* of (i) the size of the largest cluster, (ii) mean size of clusters, (iii) number of clustered tweets, (iv) the size of the smallest cluster, and (v) number of generated cluster.

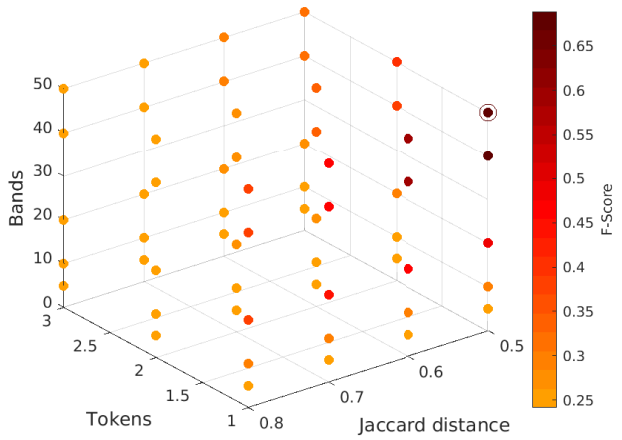

Fig. 2. Calibration phase for MinHash and LSH algorithms.

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 the authors according to the original tweet's label.

Tests were run while varying the ratio between genuine users and spammers and using different subset of features (see Fig. 3). Results showed that the best values of accuracy and f-score are obtained when *number of clusters* (f_5), *average size* (f_2), and *maximum size* (f_1) of the clusters are considered, ignoring the remaining 2 features.

Finally, in order to assess the overall performances of the automatic labeling procedure, a dataset was collected using the Twitter APIs.

As first step, the Twitter stream was queried to obtain a set of relevant tweets. Tweet collection is performed at regular intervals and exploits 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.

The dataset has been analyzed by applying the proposed procedure, that allowed to automatically detect 20 thousands legitimate users and about 2 thousands spammers. The outcomes of the labeling process are shown in Table 2. These results were compared with a ground-truth obtained by manually labeling the users we collected and the proposed approach achieved an average accuracy

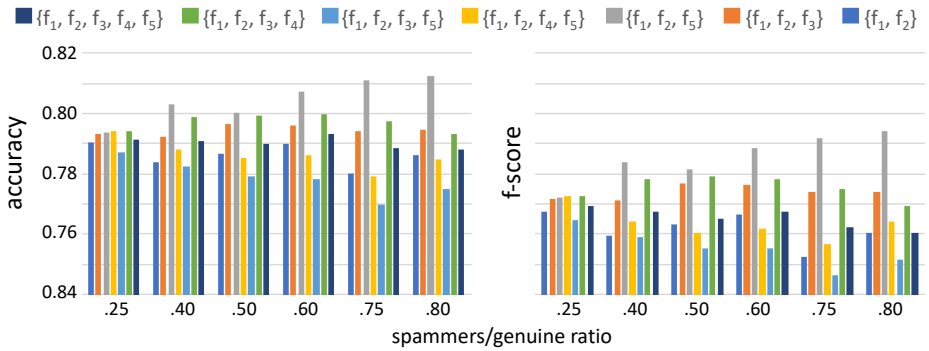

Fig. 3. Accuracy and f-score achieved while varying the ratio between *spammers* and *genuine* users in the range [25,80]. For each experiment, the following set of features were combined: size of the largest cluster (f_1), mean size of clusters (f_2), number of clustered tweets (f_3), size of the smallest cluster (f_4), and number of generated cluster (f_5).

Number of users collected	40.823
Automatically labeled as genuine	20.007
Automatically labeled as spammers	2.190
Number of tweets collected	8.010.147
Containing *urls*	2.330.558
Containing hashtags	1.640.521
Containing user mentions	4.334.056

Table 2. Output of the detection/labeling process on the dataset collected.

of about 80%. In particular we measured that the accuracy of the automatic 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 system.

5 Conclusion

In this paper we presented a system able to capture some common behaviors of Twitter spammers, i.e., the habit to share malicious *urls* and the presence of patterns in spammers' tweets.

Since the design of any new spam detection technique requires stable and annotated datasets to assess its performance, the idea is to recognize these common behaviors to provide the researchers with a tool capable of performing automatic annotation of large-scale datasets.

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.

Regarding the analysis of recurring topics and near-duplicate contents, a combination of MinHash and Local-Sensitive Hashing algorithms has been presented.

Different experiments were performed in order to determine the best set of parameters for both techniques, and to identify a set of features which permits to distinguish between spammers and genuine users.

Results showed that half of the accounts contained in the dataset can be manually labeled by means of the proposed approach with an average accuracy of about 80%. Such a result is very relevant for large-scale dataset and confirms the suitability of the proposed approach to speed-up the annotation of huge collections of Twitter data.

As future work, we want to provide an analysis tool able to find further similarities in the subset of users who need to be manually labeled. To this aim, we are investigating efficient algorithms that could allow to group similar users, analyze a few example per group, and then extend the label to the whole set.

References

1. Agate, V., De Paola, A., Lo Re, G., Morana, M.: A simulation framework for evaluating distributed reputation management systems. In: Distributed Computing and Artificial Intelligence, 13th International Conference. pp. 247–254. Springer International Publishing, Cham (2016)
2. Alsaleh, M., Alarifi, A., Al-Salman, A.M., Alfayez, M., Almuhaysin, A.: Tsd: Detecting sybil accounts in twitter. In: 2014 13th International Conference on Machine Learning and Applications. pp. 463–469 (Dec 2014). https://doi.org/10.1109/ICMLA.2014.81
3. Chen, C., Zhang, J., Chen, X., Xiang, Y., Zhou, W.: 6 million spam tweets: A large ground truth for timely twitter spam detection. In: 2015 IEEE International Conference on Communications (ICC). pp. 7065–7070 (June 2015). https://doi.org/10.1109/ICC.2015.7249453
4. Chen, C., Wen, S., Zhang, J., Xiang, Y., Oliver, J., Alelaiwi, A., Hassan, M.M.: Investigating the deceptive information in twitter spam. Future Gener. Comput. Syst. **72**(C), 319–326 (Jul 2017). https://doi.org/10.1016/j.future.2016.05.036
5. Concone, F., De Paola, A., Lo Re, G., Morana, M.: Twitter analysis for real-time malware discovery. In: 2017 AEIT International Annual Conference. pp. 1–6 (Sep 2017). https://doi.org/10.23919/AEIT.2017.8240551
6. De Paola, A., Gaglio, S., Lo Re, G., Morana, M.: A hybrid system for malware detection on big data. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops. pp. 45–50 (April 2018). https://doi.org/10.1109/INFCOMW.2018.8406963
7. De Paola, A., Favaloro, S., Gaglio, S., Lo Re, G., Morana, M.: Malware detection through low-level features and stacked denoising autoencoders. In: Proceedings of the Second Italian Conference on Cyber Security (ITASEC) (2018)
8. Fazil, M., Abulaish, M.: A hybrid approach for detecting automated spammers in twitter. IEEE Transactions on Information Forensics and Security pp. 1–1 (2018). https://doi.org/10.1109/TIFS.2018.2825958

9. Gaglio, S., Lo Re, G., Morana, M.: Real-time detection of twitter social events from the user's perspective. In: 2015 IEEE International Conference on Communications (ICC). pp. 1207–1212 (June 2015). https://doi.org/10.1109/ICC.2015.7248487
10. Gaglio, S., Lo Re, G., Morana, M.: A framework for real-time twitter data analysis. Computer Communications **73, Part B**, 236 – 242 (2016). https://doi.org/http://dx.doi.org/10.1016/j.comcom.2015.09.021, online Social Networks
11. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases. pp. 518–529. VLDB '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
12. Grier, C., Thomas, K., Paxson, V., Zhang, M.: @ spam: the underground on 140 characters or less. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 27–37. ACM (2010)
13. Kaur, R., Singh, S., Kumar, H.: 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). https://doi.org/https://doi.org/10.1016/j.jnca.2018.03.015
14. Lee, K., Caverlee, J., Webb, S.: The social honeypot project: Protecting online communities from spammers. In: Proceedings of the 19th International Conference on World Wide Web. pp. 1139–1140. WWW '10, ACM, New York, NY, USA (2010). https://doi.org/10.1145/1772690.1772843
15. Lee, K., Eoff, B.D., Caverlee, J.: Seven months with the devils: A long-term study of content polluters on twitter. In: ICWSM. pp. 185–192 (2011)
16. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge university press (2014)
17. Lua, E.K., Chen, R., Cai, Z.: Social trust and reputation in online social networks. In: 2011 IEEE 17th International Conference on Parallel and Distributed Systems. pp. 811–816 (Dec 2011). https://doi.org/10.1109/ICPADS.2011.123
18. Sedhai, S., Sun, A.: 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. pp. 223–232. SIGIR '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2766462.2767701
19. Tao, K., Abel, F., Hauff, C., Houben, G.J., Gadiraju, U.: Groundhog day: near-duplicate detection on twitter. In: Proceedings of the 22nd international conference on World Wide Web. pp. 1273–1284. ACM (2013)
20. Thomas, K., Grier, C., Ma, J., Paxson, V., Song, D.: Design and evaluation of a real-time url spam filtering service. In: Security and Privacy (SP), 2011 IEEE Symposium on. pp. 447–462. IEEE (2011)
21. Wu, T., Wen, S., Xiang, Y., Zhou, W.: Twitter spam detection: Survey of new approaches and comparative study. Computers & Security (2017). https://doi.org/https://doi.org/10.1016/j.cose.2017.11.013
22. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on neural networks **16**(3), 645–678 (2005)
23. Yang, C., Harkreader, R., Gu, G.: Empirical evaluation and new design for fighting evolving twitter spammers. IEEE Transactions on Information Forensics and Security **8**(8), 1280–1293 (Aug 2013). https://doi.org/10.1109/TIFS.2013.2267732