Detecting Spam Tweets using Character N-gram Features

Mokhtar Ashour, Cherif Salama, M. Watheq El-Kharashi
Computer and Systems Engineering Department, Ain Shams University
Cairo, Egypt
{mokhtar_ashour, cherif.salama, watheq.elkharashi}@eng.asu.edu.eg

***Abstract*—Twitter popularity made it an important and instantaneous source of news and trending events around the world. It has attracted the attention of spammers who post malicious content embedded in tweets and in their profile pages. Spammers use different and evolving techniques to evade traditional security mechanisms, and that creates the need to develop robust solutions that adapt with these techniques. In this paper, we propose using a low-level character n-grams feature that avoids the use of tokenizers or any language dependent tools. Using a publicly available dataset, we evaluate the performance of multiple machine learning classifiers with different representations of the proposed feature. Our experiments show that our approach is an enhancement over the approaches that use word n-grams from tweet tokens. We also show that our technique can detect spam tweets with low latency which is crucial in a real-time environment like twitter.**

***Keywords*—machine learning, n-grams, spam Detection, twitter**

I. Introduction

Twitter became one of the most popular social media platforms that has hundreds of millions of tweets posted everyday [1]. The platform allows users to send short messages of length 280 characters, originally 140 only [2]. Twitter serves as a real-time source of events around the world through its trending topics feature by grouping tweets that contain the same hashtag. Companies and political campaigns started analyzing tweets from users to measure the sentiment on their campaigns. Such a powerful social platform attracted the attention of spammers as it offers a better reach than traditional email spamming, they follow legitimate users, reply to their tweets and post to trending topics. During global trending events, like FIFA world cup, or presidential elections, spammers become more aggressive to exploit the high number of users on the system. At such events, manual or semi-manual techniques will not scale and the platform becomes vulnerable to spammers. Spam tweets commonly contain links to malicious websites, or advertisement campaigns. The need to detect spam tweets is the first step towards filtering them out to be able to deliver relevant content to users and make them feel safe online.

Detecting spam on twitter is not a new topic. Since the early days of twitter, researchers started to address the spam detection problem using different approaches, some works used rule-based techniques, while other works used machine learning techniques. Traditional rule-based solutions that depend on black lists of URLs or spam words are not very efficient; as spammers adapt their techniques to evade such methods. Machine learning techniques on the other hand can learn many text and content features that can detect spam tweets efficiently. Works that used machine learning to used different approaches, these approaches are categorized as account-based, graph-based, tweet-based, and hybrid approach where different methods are mixed together.

The account-based approach is addressing the problem by classifying the user account as spam or not. This approach uses historical and metadata features from the user account. Works that used this approach as in [3]–[6] used age of the account on Twitter, bio descriptions, joining date, number of liked tweets, number of retweets over time, number of lists, number of followers, number of followings, and other analysis on the most recently posted tweets (URLs ratio, unique username mentions). Some works also study accounts connected to the account under classification by measuring the distance between these accounts and intermediate accounts mentioning them in tweets. We can see that the approach requires many features to classify the account, these features need to be retrieved from twitter, or stored and updated frequently, they can also be computationally expensive if we expand the network graph.

The tweet-based approach is to automatically classify a single tweet as spam or non-spam. This approach uses content-based features such as number of hashtags, number of URLs, number of likes, number of retweets, number of replies, posting date, and the posting location if available. Text-based features are one of the most important feature sets as well, they could be n-gram analysis, POS tags, sentiment analysis, and text similarity analysis. Although the approach is very useful for real-time detection, it's not easy as it offers a limited number of out-of-the-box features compared to the account-based features.

The hybrid approaches use features from user accounts and combine them with features extracted from tweets. These approaches are reporting the best spam detection performance, so far as in [7]–[11]. The majority of these works used linear classifiers such as SVMs or Logistic regression, but most of them are reporting the best results from Random Forests Classifiers.

In this work, we are detecting spam using tweet-based features, and we are focusing our experiments on features extracted from the tweet text. We believe that other content features like retweet count and likes count could be manipulated by spammers, by creating other bot accounts to promote their spam tweets, thus appearing legitimate to many spam filtering classifiers. Most of the techniques based on text features rely on word tokens of tweets. Those techniques require robust language tokenizers to extract word boundaries, and the challenge becomes harder with languages that are rich in inflections such as Arabic, German, and Turkish. Spammers exploit this by using unexpected symbols or special characters within the tweet to confuse tokenizers (e.g. “V i d e o” or “F*R*E*E”), character-based approaches on the other hand are very robust to this kind of spam tricks.

The main contributions of this work are:

- We show how character n-grams are more robust with different spamming techniques.
- We evaluate the performance of different character n-gram feature representations (TF, TFIDF) with different supervised learning classifiers (LR, SVM, RF).
- We compare the classification and computational performance results of our approach with previous works that used word n-grams to detect spam.

We are evaluating our approach using a balanced dataset collected from 40K users. Our experiments revealed an acceptable detection rate of nearly 80% using one feature only extracted from the tweet text with a linear machine learning supervised classifier. Our evaluation results show that the proposed feature is an improvement over the word n-grams feature.

This work is organized as follows. Section II presents related works that use text-based features for spam detection. In Section III, we present our approach in detail, while in Section IV we describe how we evaluated our approach. In Section V, we discuss the results of our approach and discuss possible future works. Finally, in Section VI, conclusions are drawn.

II. Related Work

Many works approached spam filtering in general using language features. Email spam filtering attracted the attention of many researchers, and they used language features to detect spam messages from the message text analysis as in [12] and [13]. Also spam filtering techniques for online products reviews and blog comments used language features as in [14].

Few studies have focused their work on detecting spam tweets using language features, and they usually mix these features with other content or user features to obtain better results. Romo et al. [15] have used language models with Kullback-Leibler divergence measure as a feature for their classifier. To be able to compute the divergence between the probability distributions of all terms of two documents (Tweet, and web content of the URL in tweet). They used two different language models; one for short text content (suitable for tweets), and another one for long text content (suitable for web pages). Their work assumed that tweets without URLs are trusted tweets and they used these tweets to build their non-spam language model. They mark any tweet with a URL as suspicious, then they compare the language model of the page in URL with the one they built from the trusted tweets. It’s important to mention that building a language model of a web page in real-time is computationally expensive, and it’s not easy to build a flexible web parser to extract text from different types of pages.

Amleshwaram et al. [9] used a hybrid method for their detection approach, they combined account-based and tweet-based features. Although their approach is to classify accounts, they extracted features from tweets generated by these accounts. Beside their account-based features and content-features, they categorized three tweet text-based features they consider for spam analysis as content-entorpy features. They analyzed tweet’s language dissimilarity by measuring the divergence between the alphanumeric character-based probability distributions; they used three character distributions, the distribution obtained from the set of tweets for an account (test), the distribution for the English language (benign), and a uniform distribution (malicious). Another text-based feature is measuring the similarity between tweets where they applied cosine similarity between the dimensional vectors represented by unique words of tweets. They also measured the similarity between a URL content and tweet text to detect if the account is embedding a spam URL while the tweet text is making use of a trending event to promote their advertisement of malicious campaigns.

Miller et al. [16] used a combination of features from user, content, and text features. The text-based feature, introduced in their study, is based on a simple count of 95 uni-gram characters. The 95 character set was chosen from the complete ASCII set for accessibility on a standard US keyboard. They reported that character unigram gave their detection performance a considerable boost after adding the feature to their feature set. They evaluated their work using two clustering techniques, DenStream and StreamKM++ with some modifications. Although it’s unsupervised learning, training data is needed to tune the algorithm parameters.

Wang et al. [10] also used a wide range of features, user, content, and text features. They experimented using word unigram, bigram, and trigram with binary, TF and TFIDF techniques as we will present them in Section III-C and combined them with other features. They evaluated their work with five different classifiers and reported the best numbers from a mix of user and n-gram features using a random forests classifier.

Our system, as we shall see, studies the impact of using character n-gram features instead of word n-grams. We evaluate different n-grams with different techniques to represent the feature. We also evaluate the detection accuracy and computational performance of the feature with different classifiers to obtain the best results.

III. Methodology

In this section, we present the dataset that we used for evaluation, how it was collected and why we choose it, then we give an overview of our system’s architecture and how our detection approach works. Finally, we present our character n-gram feature and describe the different representation that we use with classification algorithms.

A. Dataset

For any supervised machine learning algorithm, a dataset of labeled examples is needed to be able to train the classifier, so for our spam detection problem we need a collection of labeled tweets. We could not find any publicly available dataset for spam vs. non-spam tweets. Instead, we could obtain a dataset for spammer vs non-spam Twitter user accounts with the tweets they posted. We decided to use the Social Honeypot dataset collected and labeled by Lee et al. [17]. They created 60 social honeypot accounts to attract spammers and they used Expectation-Maximization (EM) clustering algorithm to cluster the harvested users, then they manually grouped the clusters into four classes: duplicate spammers, duplicate @ spammers, malicious promoters and friend infiltrators. Furthermore, they split their dataset into legitimate users and content polluters, and provided a collection of tweets for each user account. We used the same sampling technique that [10] used to evaluate their approach. We sampled one tweet for each user in the dataset, this is to avoid having a sampling bias and overfitting. The resulting dataset contains 20,707 spam tweets and 19,249 non-spam tweets.

B. Overview

Fig. 1, shows our system's architecture which consists of two pipelines. The first pipeline works offline to train and build the classification model. First, we load the tweets labeled set, then we clean each tweet by removing all non-text data (tweet ID, user ID, tweet time) and extract the language features as we describe in the following section, then we pass the features vectors with their spam vs. non-spam label to the classifier training service. As the classifier learns different tweet patterns through the features' vectors and their corresponding labels, it starts to tune its weights to minimize the classification error. The output model is then used in the second pipeline of our system, which works online to classify new tweet to spam or non-spam classes.

C. Features Description

N-gram models are one of the most popular features in NLP classification tasks. They are preferred in many tasks because of their simplicity and scalability, where with a larger n, the model can store more context. A well understanding of their spacetime trade off, and thorough experiments are needed to build the bag of characters model by choosing a proper character count window size denoted as n. For example, the character 5-grams of the phrase "The red fox" are[1] $|The\_r|, |he\_re|, |e\_red|, |\_red\_|, |red\_f|, |ed\_fo|, |d\_fox|$. We can see that the bigger n we use, the richer context is stored, but also a bigger features space is created. The model is now built by applying the character n-grams on the tweets dataset.

For our classification model we study two representations of the character n-gram feature.

1) Term Frequency (TF): is the number of times a term (sequence of characters in our case) appeared in the tweet. By denoting the tweet as t, and the character sequence as c, tf is defined as:

$$tf(c,t) = f_c, t \tag{1}$$

[1]We use '|' to denote n-gram boundaries and '_' for a single space character.

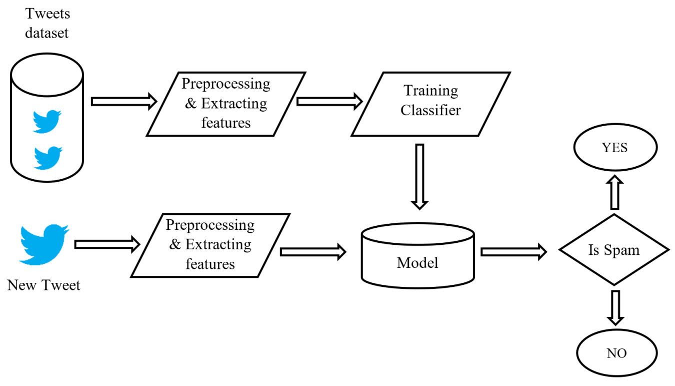
Tweets dataset
Preprocessing & Extracting features
Training Classifier
YES
New Tweet
Preprocessing & Extracting features
Model
Is Spam
NO

Fig. 1: Components of model training and spam detection pipelines.

2) TFIDF (Term frequency-Inverse document frequency): is a measure of how important a sequence of characters is to a tweet in a tweets collection. This statistic is used in many information retrieval and recommendation systems.

In order to calculate TFIDF for a sequence of characters, we need TF of that sequence and the inverse document frequency. The inverse document frequency (IDF) is a measure of how a term (sequence of characters) is common across all tweets. By denoting the number of tweets in the dataset as N, and the number of tweets where the sequence c appears as tf_c, we define the sequence idf as:

$$idf(c,t) = log(\frac{N}{tf_c}) \tag{2}$$

Now the equation of TFIDF becomes as follows:

$$tfidf(c,t) = tf(c,t) * idf(c,t) \tag{3}$$

After we calculate the TFIDF/TF for each character sequence in a tweet, the produced feature vector is the output of the feature extraction phase in both pipelines, the next step is to train the classifiers using these vectors with their corresponding tweets labels in case of the training phase (first pipeline), or to feed the vectors directly to the trained model and get the predicted class of the tweet.

IV. Evaluation

We evaluated our system using the dataset described in Section III-A. We extracted the features for each tweet and trained multiple classifiers (implemented using scikit-learn [18]): Support Vector Machines (SVM), Random Forest (RF), and Logistic Regression (LR). To be able to choose the best classifier, we evaluated the three of them with the two feature representations we extracted. Table II shows that the 1:16-grams TFIDF representation gives the best results with a logistic regression classifier, while 1:16-grams TF representation gives the best results with a Random Forests classifier.

To be able to choose the best character n-gram range for our problem, we evaluated the classifiers with a range of 1:20-grams. Fig. 2 shows the effect of applying different n-gram ranges on the F-measure of a logistic regression classifier. Also, Fig. 3 shows the effect of changing the n-gram range on the time consumed to detect 1K tweets, and it's no surprise that choosing a wider n-gram range will result into bigger feature space and more time to extract these features. As we can see in those two figures, the performance starts to saturate after 1:16-grams range and the time to detect 1K tweets is still under one second, so we decided that the 1:16-gram range is a good space-time trade off.

To compare our results with previous works, we have adopted the same technique that [19] used to evaluate their approach. They split the dataset to train/test, and used 40% of the dataset to train the classifiers (making 16K tweets) and 60% as a test set to evaluate the classifiers (making 24K tweets). Table III, shows a comparison of our results with their results, where we highlight a significant improvement in detection performance using our features versus results obtained using the word unigram, bigram, and trigram combinations that were used in their approach.

We used the standard evaluation metrics of precision, recall, and f-measure to evaluate our system and to be able to compare with previous works. Precision is the ratio of the number of spam tweets classified correctly to the total number of tweets that are classified as spam. Recall is the ratio of the number of spam tweets classified correctly to the number of tweets labeled as spam. F-measure is the harmonic mean of the precision and recall and we calculate it as follows:

$$fMeasure = \frac{2 * (precision * recall)}{(precision + recall)} \tag{4}$$

To calculate the total F-measure of our system, we apply a weighted average on the Precision, Recall, and F-measure of the two classes, where we take the number of samples in each class into consideration.

We executed all experiments on a standard modern laptop with hardware specifications similar to the machine [10] used to evaluate their approach (core i7 processor with 2.8 GHz, 16 GB of RAM, and SSD disk). We recorded the time taken for extracting the features and running the classifier. Table II, shows the time taken in seconds to classify 1000 tweets. We can see that in our case, linear classifiers (SVM and Logistic Regression) are giving better performance than Tree-based classifiers (Random Forest), which gave the best results according to many previous studies. Regardless of the classifier we are using, we can see that the feature is computationally inexpensive, and it achieves our goal of detecting spam in real-time.

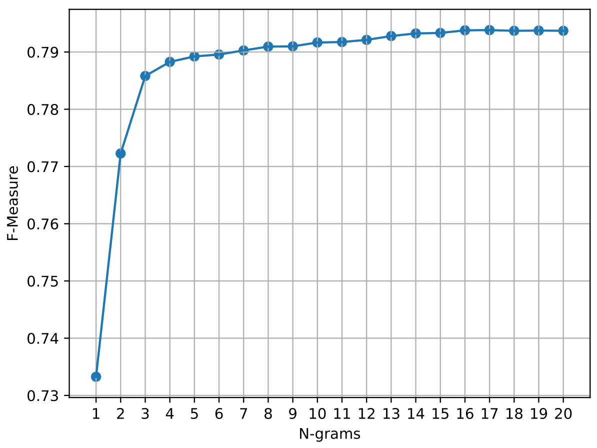

Fig. 2: Effect of increasing n-grams on F-measure.

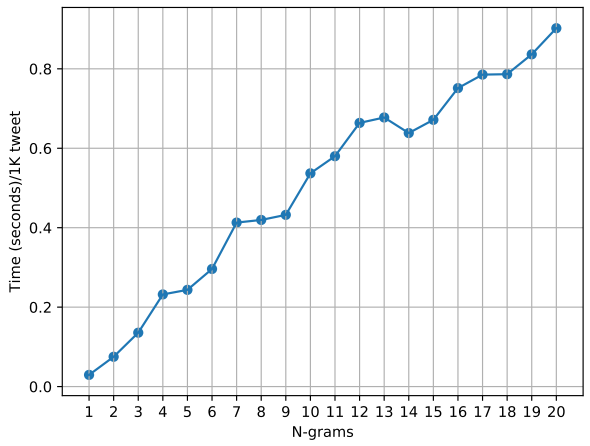

Fig. 3: Effect of increasing n-grams on detection time.

V. Discussing

Our work focused on studying the effect of using character n-gram features in spam detection in tweets. We did not mix our feature with other user and content features as many previous studies already proved the contribution of those features in obtaining better results and making the classifier robust to different spamming techniques. We believe that language features are of high importance, because other user and content-based features are more likely to be manipulated by spammers.

Because we could not find a dataset of labeled tweets for our problem, we have sampled our spam vs. non-spam dataset from a dataset that is built for spam vs. non-spam accounts. A drawback of this approach is considering that all spam accounts tweets as spam and vice versa. We believe that a manually labeled spam vs. non-spam tweets dataset is needed to evaluate tweet-based approaches accurately. That dataset could be collected from the manually spam reported tweets, then reviewed by twitter.

Another observation from our evaluation results is the relatively low precision (77.5%) of classifying a non-spam tweet correctly. We believe that dooming a non-spam tweet as spam is of higher penalty than classifying a spam tweets as non-spam, thus we plan to work on enhancing the precision of classifying non-spam tweets. We plan to add different tweet content-based features in our experiments as we believe it could lower the false negative rate.

VI. Conclusion

In this work, we addressed the problem of spam detection in tweets using language features, which differs from most of previous detection techniques that focus on detecting the

TABLE I: Classifiers performance with two features representations

Feature Representation	**Character 1-16 grams TF**			**Character 1-16 grams TFIDF**		
Classifier	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
SVM	0.7540	0.7538	0.7538	0.7785	0.7783	0.7784
Random Forests	0.7903	0.7841	**0.7836**	0.7879	0.7807	0.7801
Logistic Regression	0.7768	0.7760	0.7760	0.7945	0.7937	**0.7938**

TABLE II: Performance computation for 1K tweets

Classifier	**Character 1-16 grams TF**	**Character 1-16 grams TFIDF**
SVM	0.7489	0.6986
Random Forests	1.0032	0.8104
Logistic Regression	0.7267	0.7514

TABLE III: Comparison with previous work

System	**Classifier**	**Feature representation**	**Precision**	**Recall**	**F-measure**
Wang et al. (2015)	RF	Word 1-2 grams TF	0.783	0.767	0.775
Wang et al. (2015)	RF	Word 1-2 grams TFIDF	0.784	0.765	0.775
Our system	RF	Character 1-16 grams TF	0.790	0.784	0.784
Our system	LR	Character 1-16 grams TFIDF	**0.795**	**0.794**	**0.794**

user account. We focused our experiments on the impact of character n-gram feature in spam detection. First, we described the feature and showed how it could be effective with spammers' techniques that confuse common language tokenizers. Secondly, we evaluated our feature using a manually labeled dataset with three classifiers, Random Forests, Linear SVM, and Logistic Regression, and showed that character n-gram is an enhancement over the common word n-gram techniques used in previous studies. We evaluated the classification performance of different character n-gram ranges with TF and TFIDF techniques and compared the results with previous works that used the same techniques with word n-grams. Finally, we evaluated the computational performance of our system and showed that we can classify 1K tweets under one second by using the proposed features.

As a future work, we plan to extend this study to other languages to make use of the powerful feature of character n-grams where they do not require any language dependent tools. Also, we plan to study the impact of using character level embeddings in our problem as it has been used with tweets sentiment analysis and similar short text problems. The embeddings when used with different deep learning algorithms produced good results as in [20] and [21]. Finally, we believe that using active learning in spam detection problem by taking user feedback and feeding it into the system as new labeled data would keep enhancing the system and make it robust against new and evolving spammers techniques.

References

[1] "Twitter usage statistics." [Online]. Available: http://www.internetlivestats.com/twitter-statistics [Accessed: 2018-11-15]

[2] S. Perez, "Twitter officially expands its character count to 280 starting today," *Techcrunch*. [Online]. Available: https://techcrunch.com/2017/11/07/twitter-officially-expands-its-character-count-to-280-starting-today [Accessed: 2018-11-15]

[3] G. Gee and H. Teh, "Twitter spammer profile detection," Stanford University, CA, 2010.

[4] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots+ machine learning," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 435–442.

[5] P.-C. Lin and P.-M. Huang, "A study of effective features for detecting long-surviving twitter spam accounts," in *Advanced Communication Technology (ICACT), 2013 15th International Conference on*. IEEE, 2013, pp. 841–846.

[6] J. Song, S. Lee, and J. Kim, "Spam filtering in twitter using sender-receiver relationship," in *International workshop on recent advances in intrusion detection*. Springer, 2011, pp. 301–317.

[7] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proceedings of the 26th annual computer security applications conference*. ACM, 2010.

[8] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, vol. 6, no. 2010, 2010, p. 12.

[9] A. A. Amleshwaram, N. Reddy, S. Yadav, G. Gu, and C. Yang, "Cats: Characterizing automation of twitter spammers," in *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*. IEEE, 2013.

[10] B. Wang, A. Zubiaga, M. Liakata, and R. Procter, “Making the most of tweet-inherent features for social spam detection on twitter,” *arXiv preprint arXiv:1503.07405*, 2015.

[11] 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 *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 7065–7070.

[12] I. Kanaris, K. Kanaris, I. Houvardas, and E. Stamatatos, “Words versus character n-grams for anti-spam filtering,” *International Journal on Artificial Intelligence Tools*, vol. 16, no. 06, pp. 1047–1067, 2007.

[13] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos, and P. Stamatopoulos, “Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach,” *arXiv preprint cs/0009009*, 2000.

[14] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, “Finding deceptive opinion spam by any stretch of the imagination,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 309–319.

[15] J. Martinez-Romo and L. Araujo, “Detecting malicious tweets in trending topics using a statistical analysis of language,” *Expert Systems with Applications*, vol. 40, no. 8, pp. 2992–3000, 2013.

[16] Z. Miller, B. Dickinson, W. Deitrick, W. Hu, and A. H. Wang, “Twitter spammer detection using data stream clustering,” *Information Sciences*, vol. 260, pp. 64–73, 2014.

[17] K. Lee, B. D. Eoff, and J. Caverlee, “Seven months with the devils: A long-term study of content polluters on twitter.” in *ICWSM*, 2011, pp. 185–192.

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

[19] A. H. Wang, “Don’t follow me: Spam detection in twitter,” in *Security and cryptography (SECRYPT), proceedings of the 2010 international conference on*. IEEE, 2010.

[20] C. dos Santos and M. Gatti, “Deep convolutional neural networks for sentiment analysis of short texts,” in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 69–78.

[21] S. Vosoughi, P. Vijayaraghavan, and D. Roy, “Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder,” in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 1041–1044.

[22] A. T. Kabakus and R. Kara, “A survey of spam detection methods on twitter,” *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 3, 2017.

[23] C. Yang, R. C. Harkreader, and G. Gu, “Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers,” in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2011, pp. 318–337.

[24] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.