Make Social Networks Clean Again: Graph Embedding and Stacking Classifiers for Bot Detection

Kirill Skorniakov
Ivannikov Institute for System Programming of the Russian Academy of Sciences
Moscow Institute of Physics and Technology (State University)
Moscow, Russia
kirill.skorniakov@ispras.ru

Denis Turdakov
Ivannikov Institute for System Programming of the Russian Academy of Sciences
National Research University Higher School of Economics
Moscow, Russia
turdakov@ispras.ru

Andrey Zhabotinsky
Ivannikov Institute for System Programming of the Russian Academy of Sciences
Lomonosov Moscow State University
Moscow, Russia
zhabotinsky@ispras.ru

Abstract

The paper introduces a novel approach to the detection of social bots using ensembling of classifiers. We also studied the impact of different feature sets and demonstrated the power of graph embedding which is underused by the existing methods. The main contribution of this work is a creating of a stacking based ensemble, which effectively exploits text and graph features. Empirical evaluation proved the effectiveness of the proposed method for bots detection and showed improvement in comparison to existing solutions by 4-9 points of AUC.

1 Introduction

Online social networks (OSNs) are an important part of life for many people. More than 2 billion people use social network services. Unfortunately, online social networks like many other technologies provide opportunities for illegal and undesirable activities. Thus OSNs can be used to spread spam, phishing links, fake news. In addition, there are many malefactors and fraudsters who extort money from users or engage in illegal advertising. They disturb and defraud upstanding users. That is why the administration of social networks tries to find and block their profiles.

Effective automatic methods are required to search for bots on the scale of the entire social network. Many papers addressed this problem in recent years [10, 13, 18, 1, 8]. Most of them are based on supervised machine learning and produce valuable results. Authors use features extracted from information about particular profile available in most OSNs such as the publication of posts and comments, the formation of non-directional (friendship) and directed (subscription) links. As far as we know, there are no works that use the global network structure for this purpose. In this paper, we correct this omission with the aid of graph embedding.

Any development of a supervised classifier for bot detection meets several difficulties. First of all, it's hard to determine the definition of a "bot" and receive a labeled dataset. Another problem is the creation of a qualitative classifier that can effectively use and combine text, graph and other types of data. In this work, we explore and utilize the existing solutions for the first problem and focus on improving solutions for the second problem.

Many good algorithms that use different feature extraction methods were developed recently. It seems reasonable to combine them in order to achieve state-of-the-art result and stacking of algorithms can help achieve a quality gain in this task.

To summarize, we make the following contributions:

- We applied graph embedding techniques to extract additional features from the entire network;
- We also created an efficient stacking based bot classifier, which combines graph and text information.

The rest of the paper is organized as follows. In Section 2 we describe existing solutions. Section 3 is devoted to our stacking-based classifier. Section 4 presents our experiments. At the Section 5 we summarize and conclude.

2 Related Work

2.1 Bot Detection Labels

An important issue is to collect a labeled dataset to detect bots using machine learning algorithms The main difficulties arise from the definition of the concept of a bot. However, despite the ambiguity of the definition, there are ways to obtain a rather precise labeling of bot dataset.

According to [10] all labeling techniques can be categorized into three groups. They are:

- Manual annotation;
- Lists of suspended users – users blocked by the social network administration;
- Honeypots [8] – bots created by researches to lure other bots.

We use the second approach because it requires the least human resources for the markup and utilizes an underlying information about the rules used by social network administration which is unobtainable to external researchers (for example, the number of complaints about the user).

2.2 Bot Detection Features

A wide range of various features can be extracted from a social network and used for bot detection. Based on [10, 13, 18, 1] we can group these features in the following way:

- Text features which include the number of hashtags, links, geo-tags, words from spam list, statistics of posts sentiment and topics;
- User-profile features usually imply the username, the number of friends, subscribes, photos, audios, retweets;
- Time features include statistics of user online time, user publications time;
- Graph features involve information extracted from friendship and subscribes graphs such as PageRank and centrality.

Fred Morstatter et. al. [10] used topics distribution, obtained by the Latent Dirichlet Allocation (LDA) algorithm on tweets. Kyumin Lee et. al. [8] utilized statistics of a user's followers and tweets. Onur Varol et.al. [15] proposed framework with many different features. They are statistics of emotions, pos tags; posting time features, simple statistics of retweets, mentions, and hashtag co-occurrence networks. We will use works [8, 10] as baselines. Unfortunately, we can't directly compare with the [15] because they used many tools and features specific in English, but in our samples, we have many different languages.

To the best of our knowledge, the researchers did not apply graph embedding techniques to the entire social graph to improve their solution to bots detection.

2.3 Graph Embedding

In the classical approach to building a feature vector for objects, features are usually invented by experts in the field. This approach has several limitations. First of all, for each new area, an involvement of experts leads to additional expenses for development. In addition, features created by them could be computationally complex. Therefore, methods of learning representations, which automatically obtain qualitative representations of small dimensions became popular in recent years. Such representations are usually called "embedding" (word embedding, graph embedding, etc.). There are a large number of such methods for graphs. DeepWalk [12], Node2vec [4], and LINE [14] are the most popular of them due to the linear complexity of the number of edges or vertices.

Since in our experience the quality of the above algorithms does not differ much for unweighted large graphs, we used in our research LINE because of its good computational speed.

In the LINE two probability models of the appearance of edges in the graph were proposed. The first model, which preserve first-order proximity maximizes the joint probability of the observed edges $\{(v_i, v_j)\}_{i,j}$ in accordance with the following model:

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^T \vec{u}_j)},$$

where $\vec{u}_i$ – vector representations of nodes.

In the second model, vertices have two different representations (as in word2vec [9] model): $\vec{u}_i$ when vertex treated as vertex itself and $\vec{u}'_i$ – when vertex treated as the context of other vertices. Then joint link probability would be $p(v_i, v_j) = p(v_i)p_2(v_j|v_i)$, where

$$p_2(v_j|v_i) = \frac{\exp(\vec{u'_j}^T \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u'_k}^T \vec{u}_i)}.$$

The hidden parameters of these models are vector representations of the vertices. These parameters are

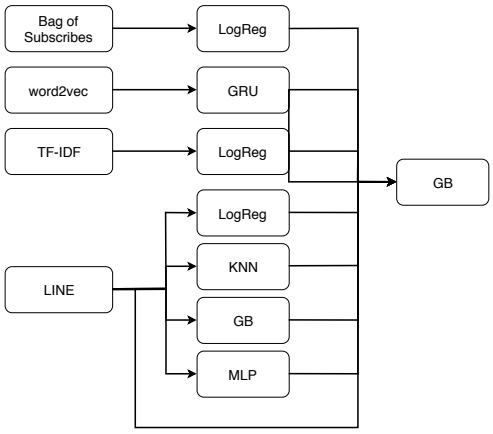

Figure 1: Proposed scheme of stacking

obtained by minimizing the Kullback-Leibler divergence between the model and observed distributions (which is equivalent to maximizing the likelihood). To accelerate calculations, the method of negative sampling [9] is used.

3 Stacking for bot detection

In this section, we turn to the construction of a classifier for prediction of bots. We use three types of attributes for each user: friendship graph, subscription information, and user's texts. We produce various transformations of these attributes to obtain a great set of vector representations of a user.

Then there are two main options for exploring different feature spaces: combine these views with one large classifier or train a separate classifier on each of them and combine these classifiers with ensemble techniques. The first option considers relations between features in various spaces which could improve the predictive power of a model. But at the same time, such classifiers have less flexibility and a tendency to overfitting. By flexibility, we mean the ability to use the classifier or its parts in the case when for some features don't exist for certain users or reusability of parts of the classifiers in the new domain. In addition, the effectiveness of the method of ensemble algorithms was repeatedly proved in the data analysis competition and papers. Therefore, we chose the second option.

For each of the three types of attributes, we build their transformations into vector representations and train on them different classifiers.

3.1 Text classifiers

As one text representation we use the concatenation of bags of words and char n-grams representations, where $n = (3, 4)$. Resulting feature vector has $|D| + |D_{3char}| + |D_{4char}|$ dimensions, where $|D|$ — size word dictionary, $|D_{3char}|$, $|D_{4char}|$ — sizes of 3- and 4-grams dictionaries. Then TF-IDF transformation was applied to this feature vector and logistic regression was used for classification.

Table 1: Bot Datasets

Social Network	Active	Banned
VKontakte	158806	21263
Twitter	69602	7886

Another classifier was GRU recurrent neural network [2] that receives word embeddings created by well-known word2vec model [9] as input. We used word vectors with 300 dimensions preliminary trained on the our large corpus (3.3GB) of informal text collected from online social networks.

3.2 Subscribes classifier

Information about subscribes we handle the following way:

1. Select largest N groups by number of subscribers;
2. Each user is represented as "bag of subscribes" – by the selected at previous step groups.

"Bag of subscribes" is a binary vector with information about subscribes. The i-th element of this vector is 1 if the user is subscribed to the i-th group and 0 otherwise. Length of final vector is N.

Then we train logistic regression on this feature vector with $N = 10000$.

3.3 Graph classifiers

For friendship analysis we use graph embedding features with size $d = 100$ obtained from LINE as input to different classifiers:

- Multilayer perceptron (MLP)
- K nearest neighbours with cosine distance (KNN)
- Gradient Boositng Classifier (GB)
- Logistic Regression (LogReg)

Machine learning algorithms were implemented with scikit-learn [11], lightGBM [6] and keras [3].

3.4 Stacking

There are several common ways to combine classifiers predictions. The most popular are the weighted average and stacking. In case of weighted average resulting prediction is a convex combination of all classifiers scores.

The main idea of stacking [16] is to use predictions of existing classifiers (which are called "first-layer" classifiers) as new features. Then a new classifier, called a meta-classifier, is trained on them.

We append vector representations of the vertices of the friendship graph as the most "powerful" features to the predictions of classifiers of the first level. Then we use gradient boosting as a meta-classifier. The scheme of the proposed method is shown in Figure 1.

4 Evaluation

In this section, we empirically evaluate the proposed method on two real-life datasets and compare our results with existing solutions.

4.1 Twitter Dataset

There are several papers, which share their Twitter-based datasets ([15, 8]) [1] [2]. Unfortunately, these datasets don't contain followers graphs, because this data can't be easily collected due to Twitter's API limit. Also, we can't obtain any information from suspended accounts. That's why we construct dataset in the following way: we use tweets from [17] [3], graph from [7] [4] and relations between *sceen_name* and *user_id* (for suspended users) from [5]. The resulting dataset consists of 77488 users that are present in all these three datasets.

We used Twitter API to determine the suspended user's. Users who have been suspending received a label of 1, the rest received a label of 0.

4.2 VKontakte Dataset

In order to show that our method is applicable on different social networks, we also collect data from VKontakte (Russian social network similar to Facebook) that had friendly open API with fewer restrictions than in most OSNs. We use only public information, such as user's posts and comments in open groups, their friendship information, group subscribes.

Data is downloaded using VKontakte API. We use information about the status of users as labels. Users can have one of the following statuses: "active", "deleted", "banned". Users with the "deleted" status have deleted their accounts themselves. Users with the "banned" status were blocked by VKontakte administration. We can't obtain any information about non-active profiles due to restrictions of VKontakte.

For obtaining labels for VKontakte dataset we use the following process:

1. All users profiles from the social network are collected;
2. Profiles with the "deleted" and "banned" statuses are excluded from the sample;
3. Statuses of all remaining accounts are recollected;
4. Users with the "banned" status after recollection receive a label "1";
5. Users with the "active" status receive a label "0".

We use data collected from 2 until 16 November 2017 for our experiments [5].

Of all users, we select users with more than 10 friends and texts in the open groups for the last month. The restriction on the number of friends is introduced to accelerate the computation of vector representations. The second restriction is necessary in order to use text attributes in the final algorithm. Note that both these restrictions only affect the speed of the experiment, because if one of the attributes (text or graph of friendship) is absent, the user can be classified by another available attribute using only one classifier from the ensemble. The resulting graph contains 110 million of vertices and 7 billion of edges.

To reduce the required computational resources for machine learning algorithms we select a small sample from the original dataset. We take all banned user among the selected above as positive labels and 5% of active users as negative labels. The parameters of the final dataset are shown in the Table 1. Note that the algorithms of the graph embedding algorithms were still trained on the whole friendship graph for obtaining good node representations.

4.3 Metric and Baselines

As a measure of quality, we use the area under ROC-curve (AUC), that is a common metric for binary classifications tasks.

As a first baseline algorithm for comparison, we took the results of the detection of bots from the paper by Zegzhda et.al. [18]. The choice of the paper is justified by using the same task, quality measures, and social network. We also used Twitter bot detection algorithms BoostOR [10] and Boosting [8]. Our implementation of the Boosting algorithm didn't use one feature — dynamic of followers number, because we don't have this information in our datasets. For

[1] https://botometer.iuni.iu.edu/bot-repository/datasets.html

[2] http://bit.ly/asonam-bot-data

[3] http://snap.stanford.edu/data/bigdata/twitter7/tweets2009-09.txt.gz

[4] https://snap.stanford.edu/data/twitter-2010.html

[5] VKontakte dataset is available at http://talisman.ispras.ru/datasets/

[6] We did not re-implement the method and only give the result published by the author [18]

Table 2: Bot detection results

Transformation	Classifier	VK	Twitter
tf-idf	LogReg	0.714	0.632
Word2vec	GRU	0.687	0.592
LINE	GB	0.779	0.639
	LogReg	0.760	0.629
	KNN	0.756	0.623
	MLP	0.769	0.638
"Bag of subscribes"	LogReg	0.692	-
first-layer classifiers	Weighted Average	0.802	**0.652**
first-layer classifiers + LINE	GB	**0.820**	**0.652**
Zegzhda et al. [18]	MLP	0.73 [6]	-
LDA	BoostOR [10]	0.659	0.613
Lee et al. [8]	Boosting [8]	0.637	0.605

performance estimation, we do standard 5-fold cross-validation.

4.4 Results and Discussions

The results of the experiments are shown in Table 2. It's clearly seen that proposed method based on stacking different classifiers outperform existing approaches by 4-9 points of AUC. It is worth paying attention to the features obtained from graph embedding techniques. LINE provides powerful user representations. It even allows achieving good quality with single classifiers. We believe that the main reason for this is that the graph structure is the most complex characteristic of a user account. Thus, the creation of a bot with a friendship graph similar to a normal user is a complex and time-consuming task and the most bot makers don't do this.

Results also show that stacking of first layer classifiers with graph embedding features allows boosting the best single classifier scores by 1-4% of AUC. We consider that the relatively poor results of all algorithms on the Twitter dataset are caused by peculiarities of its construction. There was more than 8 years between the collection of user's texts and graphs and the moment of the collection of labels. Also, to the best of our knowledge, Twitter does not let you know if the user deleted his profile or it was blocked.

5 Conclusion

This paper presents an ensemble approach to bots detection problem. We showed that graph embedding techniques can be used to obtain powerful features to this task. We proposed stacking algorithm which effectively combines text- and graph-based classifiers and achieves the best score on the tests. Experimental results on real-life datasets show the effectiveness of the proposed method and its applicability to the analysis of the different social networks. Our method improved existing solutions by 4-9% of AUC score.

5.1 Acknowledgements

This work is funded by the Minobrnauki Russia (grant number: 14.604.21.0199 (id:RFMEFI60417X0199)).

References

[1] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, page 12, 2010.

[2] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[3] F. Chollet et al. Keras, 2015.

[4] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[5] N. O. Hodas and K. Lerman. The simple rules of social contagion. *Scientific reports*, 4:4343, 2014.

[6] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[7] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. AcM, 2010.

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

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[10] F. Morstatter, L. Wu, T. H. Nazer, K. M. Carley, and H. Liu. A new approach to bot detection: striking the balance between precision and recall. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 533–540. IEEE Press, 2016.

[11] F. Pedregosa et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[13] V. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, K. Lerman, L. Zhu, E. Ferrara, A. Flammini, and F. Menczer. The darpa twitter bot challenge. *Computer*, 49(6):38–46, 2016.

[14] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[15] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, A. Flammini, V. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, et al. Online human-bot interactions: Detection, estimation, and characterization. *Comm. ACM*, 59:7, 2016.

[16] D. H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[17] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186. ACM, 2011.

[18] P. Zegzhda, E. Malyshev, and E. Y. Pavlenko. The use of an artificial neural network to detect automatically managed accounts in social networks, 2017.