Bot detection by friends graph in social networks

Maxim Kolomeets[*], Andrey Chechulin[†], and Igor Kotenko[‡]
St. Petersburg Federal Research Center of the Russian Academy of Sciences
St. Petersburg, Russia
{kolomeec, chechulin, ivkote}@comsec.spb.ru

Received: March 1, 2021; Accepted: April 28, 2021; Published: June 30, 2021

Abstract

In this paper, we propose a machine learning approach to detecting malicious bots on the VKontakte social network. The advantage of this approach is that only the friends graph is used as the source data to detect bots. Thus, there is no need to download a large amount of text and media data, which are highly language-dependent. Besides, the approach allows one to detect bots that are hidden by privacy settings or blocked, since the graph data can be set indirectly.To do this, we calculate graph metrics using a number of graph algorithms. These metrics are used as features in classifier training.The paper evaluates the effectiveness of the proposed approach for detecting bots of various quality - from low quality to paid users. The resistance of the approach to changing the bot management strategy is also evaluated. Estimates are given for two classifiers - a random forest and a neural network.The study showed that using only information about the graph it is possible to recognize bots with high efficiency (AUC-ROC greater than 0.9). At the same time, the proposed solution is quite resistant to the emergence of bots having a new management strategy.

Keywords: Bot detection, Social networks, VKontakte, Graph algorithms, Random forest, Neural network

1 Introduction

Bot detection is one of the key mechanisms in protecting social networks. If earlier, only Facebook could be called a social network, today many services add to themselves the functions of social networks. Some non-obvious examples – online stores that have product reviews and social ratings of sellers, banks that have messages function and friendslists for fast money transferring, news platforms – that allows one to subscribe to authors and comments on their work. All of this creates a social relationship that can be attacked: an attacker can understate the rating of a competitor in an online store, can pretend to be someone to fraud, or simply spread rumors and misinformation in comments on a news site. It can be expected that social functions will continue to be implemented in various types of services. Therefore, threats from bots will become more widespread.

There are many threats on social media, including password leaks, use of private data by third-party companies, compromise of personal correspondence, etc. But we focus only on those threats that can be implemented using bots, which include: fraud, promotion of harmful or censored information, and rating manipulation. Malicious bots pretend to be real people because it is a key component of user trust, that is necessary for threats implementation.

[*]Developing an idea and conducting experiments

[†]Developing an idea and provided a lot of suggestions

[‡]Corresponding Author: St. Petersburg Federal Research Center of the Russian Academy of Sciences, 39, 14th Linija, St. Petersburg, Russia, Email:ivkote1@mail.ru, Tel:+79217504307

Due to its great relevance, the direction of research on detecting bots is well developed. The main contribution to the development of this area is made by social networks themselves, law enforcement agencies of various governments, research institutes, and private organizations (analytical agencies and associations of journalists). Most of the proposed bot detection methods are based on the analysis of the profile, text, and other generated content.

In this paper, we propose a graph-based approach for bots detection. This approach can be described by one key feature which underlines the novelty of the paper: a bot is identified exclusively by the graph of its friends. This has several advantages: (1) there is no data that depends on the language, (2) it is more difficult for the bot to simulate the structure of the graph in comparison with profile data, (3) even if the bot's profile is hidden - one can get its graph indirectly.

This paper consists of the following sections: (2) **State of the art** – where we described existing approaches for bot detection and methods of forming graphs, (3) **Bots detection based on graph of friendslist** – where we describe the proposed approach that includes graph generation, feature construction, and machine learning decision making, (4) **Evaluation** – where we conduct experiments that include dataset generation, feature analysis and effectiveness of the proposed approach in bot detection, (5) **Discussion** – where we analyzed results, (6) **Conclusion** – where we summarizing the results and present plans for future work.

2 State of the art

In this section, we review the current state of bot detection research. We consider its main components: decision-making approaches, bots features, and data sources.

The three main classes of detection algorithms (based on taxonomy [1, 37]) are:

- Supervised learning. These algorithms allow training classifiers on pre-labeled data. Usually, researchers try to solve a binary classification problem when it is necessary to distinguish a bot from a real user (this is the most common task [9, 37, 1]). But it can also be a multi-class classification task, for example, when one goes to detect users and several types of bots [45].

- Semi-supervised learning. Algorithms that train machine learning models using datasets that have only a small labeled part. This approach is very promising because the size and quality of data labeling is almost the main problem for researchers. For example, in [10, 42] semi-supervised classification techniques for bot detection is used.

- Unsupervised learning. Algorithms that try to cluster accounts so they don't need labeled data. We find such an approach interesting because it allows one to work with a group of accounts at once and not separately. Moreover, such methods are not related to the quality of the datasets. A good example of unsupervised learning based bot detection is DeBot [5] and [32]. However, it is unclear whether such approaches can detect more sophisticated bots.

Which particular algorithm is the best to use is difficult to predict. But as follows from the reviews [37, 20], the most popular is Random Forest. Other methods include AdaBoost, Supported Vector Machine, K-Nearest Neighbor Search, and various types of neural networks (Long Short-Term Memory, Convolutional Neural Networks, Back Propagation Neural Networks).

Nevertheless, to a greater extent, the efficiency of algorithms depends more on the features used, the quality of the dataset, and the problem being solved. Therefore, researchers usually test several algorithms at once, choosing the one that gives the best result.

Another important aspect is feature construction from data. It is possible to use four "families" of methods for feature construction. We have compiled this classification mainly based on papers [1, 9]:

(a) Statistical - methods [6, 29, 11, 19, 3, 18, 9] that are based on searching for features in distributions. They are most common for feature construction since a very large amount of data is represented by distributions (distribution of parameters of friends, text, etc.).

(b) Network science - methods [34, 11, 46, 1] that are based on the analysis of graph structures that can be formed by user or content of the social network. Network science algorithms can be applied for various types of graphs to calculate centrality measures, such as betweenness centrality, eigenvector centrality, clustering coefficient, and others. For large graphs, calculations on the GPU can be used [28, 25]. One can also use graph visualization [23] (a popular method for manual detection).

(c) Machine learning methods are used very often, because on social networks a lot of data is presented in non-numerical form - texts, images, videos, audios, etc. The machine learning approach to feature extraction can be broken down into several more approaches:

   - Video, image, or audio recognition. Analyzing media information can be very valuable because it is extremely difficult for bots to generate.

     For example, if a bot account is created from scratch, the bot needs to take profile photos somewhere. If a bot steals a photo from another user [36], one can find the original profile using a facial recognition service [41, 4]. In addition, methods [30, 33, 17] are being actively developed for detecting photographs that were generated [47] by neural networks .

     One can also use ready-made solutions [12] to determine what is shown in the photo and video. Object types can also be used as features.

     Audio messages can be converted into text [22] and analyzed using methods for text analysis.

   - Natural language processing is used for text analysis. There are many articles devoted to text-based detection, largely because the most popular social network for bot researchers is Twitter (and there is almost nothing on Twitter except text).

     Feature extraction based on the text analysis can be various. For example, one can try to recreate numerical features of a user's profile (such as gender, age, and so on) based on the written text [16].

     Another popular approach [48, 2, 40] is to use word embeddings - vector representations of text (typical solutions are word2vec algorithm [31] and pre-trained GloVe word vectors [38]).

     Another well developed approach is to use sentiments [8, 15, 7] (different characteristics of emotions and opinion) as features.

   - Learning on graphs. As with text, graph structures can also be converted to vector space. For this, algorithms such as node2vec [14] and DeepWalk [39] are used. By transforming the structure of the graph into a vector, it can be used as a set of features [21, 43].

To review and classify data sources, we relied on several works devoted to bot detection [7, 37, 13, 36, 1], and especially on the papers [9] devoted to bot features on Twitter. We found that it most fully describes the classes of data sources and it is not difficult to interpolate these classes from Twitter to other social networks. There are 5 big groups of data sources:

(a) Account-based – data from account's home page. Home pages vary depending on the social network. They usually include the name, city, hobbies, and other general information that helps describe the user.

(b) Adjacent account-based – distribution parameters of account-based, that one can extract from adjacent accounts (accounts that are somehow connected with the analyzed account).

(c) Text and Media – data that one can extract from the generated content. The content can include text in posts or comments, videos, photos, polls, emojis, live streams, wish-lists, gifts, and so on.

(d) Graphs – data that can be obtained by analysis of graphs of accounts (the vertex is account) or graphs of content (the vertex is text/media/etc.).

(e) Temporal - data on the dynamics of actions performed. These can include time-series of publications, frequency of writing comments, etc.

Most of the proposed methods for detecting bots are based on the analysis of user-generated content: text, media information, profile fields, etc. Besides, most of the work on detecting bots is devoted to Twitter (in which text is the main source of data). As a consequence, such methods of analysis can be very language-dependent, which can vary greatly in different countries or social groups.

In our approach, we propose to use only graphs for analysis as a data source. Therefore, we will separately consider the approach to forming a graph that describes the bot.

When discussing the analysis of social network graphs, researchers usually mean the analysis of social relationships between users. Such social relations can be expressed in different ways depending on the functionality of the social network. This can be friendship, kinship, a community with similar interests, event participants, etc. In this paper, we do not propose a taxonomy of social relations, the development of which is more a problem for sociologists. But we are more interested in what types of graph structures can form in social networks. Behind all diversity of social networks, there are three functions, each of which forms a different structure of graphs:

(a) Friendship. By adding friends, a person strives to reproduce their real circle of acquaintances. As a result, many overlapping communities are formed (small world structure). Examples of social networks with such functions are Facebook (add friend), Twitter (subscribe), or LinkedIn (connect).

(b) Participation. The participation function is needed to indicate a connection of the user with some source of information. As a result, a bipartite graph is formed: on the one hand, users, and on the other hand, sources of information. Obvious examples are YouTube (users subscribe to channels), Instagram (users like photos), VKontakte (users participate in events).

(c) Discussion. In the discussion, users can respond to each other, forming a thread of messages. For example, comments on Facebook, threads of Twits on Twitter, or messages on public chats on Telegram.

These functions form graphs as shown in Figure 1. Mark Newman in his paper [35] has already argued that such graphs can be transformed to obtain a graph of social relationships between people (not between people and content). The transformation of such graphs can be easily accomplished using basic graph database tools, as shown in [26, 25].

There are two key points to keep in mind:

(a) Social connections that are formed by these functions are not necessarily public. For example, in messengers (ex. WhatsUp), the *friendship* function is not public - one cannot see another person's friendslist. But one can indirectly establish a user's *subscription* to a public chat by finding this person in the list of chat participants. And one can establish this person's *discussion's* messages only within one public chat.

(b) If friendship displays the social connection between users directly, then subscription and discussion display social connections indirectly (Figure 1). For example, we can connect all *subscribers* of one YouTube channel with a fully connected graph of social connections, as shown in the center of Figure 1. One can also link the participants in the same *discussion* who are responding to each other, as shown on the right side of Figure 1.

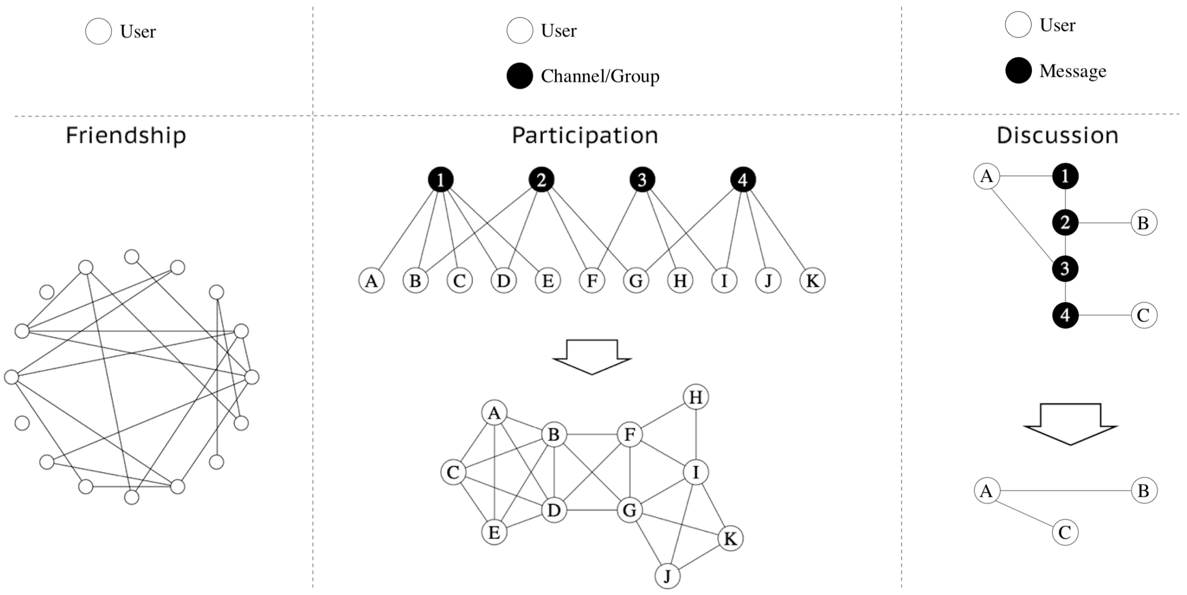


Figure 1: Graphs that can be created by friendship, participation, and discussion functions. Participation and discussion graph can be transformed into a graph of users.

Indirect social connections (subscription and discussion) are more difficult to analyze because they provide distorted information. However, for some social networks, the analysis of indirect social connections is the only possible solution (ex. messengers). Such an approach to the formation of graphs allows one to obtain a significant variety of data for analysis.

3 Bots detection based on graph of friendslist

The bot detection approach we propose is based on machine learning and is a classic pipeline typical for data science tasks. We divided the approach into three sequential tasks - getting the graph, feature construction, and training the model.

3.1 Getting the graph

The proposed approach uses only one data source to detect the bot - the graph of friends. Getting a graph of friends has two parts.

Graph traversal. The first step is to find friends, connections between them and form a graph. Data collection can be described using a traverse function with a depth of 2. At depth 1, one can get a *list of friends* and associate them with the analyzed account. At depth 2 we get *friends of friends*, and we can associate them with the *list of friends* we got at depth 1. Visually, the vertices and edges of the graph that we get at the 1st and 2nd depth of the traverse are shown at the top of Figure 2.

The API complexity (the number of API requests) of getting data from a social network depends on the size of the set of vertices for which the traverse is performed. For 1st depth traverse API complexity is 1 (as there is only one vertex) and for the 2nd depth traverse API complexity equal to the number of vertices obtained at depth 1 traversal (number of friends). Thus, overall API complexity is:

$$API\ complexity = 1 + number\ of\ friends$$

Graph generation. Based on the friendslists, one can generate several graphs of friends, which will have different topologies. The resulting graph has 4 vertex layers (bottom of Figure 2) - (1) the analyzed account, (2) the list of friends, (3) mutual friends of friends, and (4) not mutual friends of friends.

Layer 2 (friendslist) must be present in all types of graphs, while other layers can be included or filtered:

- The analyzed account. It is connected with all vertices from layer 2, so its inclusion/filtering may affect the results of algorithms using path calculation.
- The mutual friends of friends. They allow one to establish indirect friendships for vertices from layer 2 through mutual friends. Inclusion/filtering may affect the results of algorithms using path calculation and the number of neighboring vertices.
- Not mutual friends of friends. It allows one to determine the number of friends for vertices from layer 2. Inclusion/filtering may affect the results of algorithms using the number of neighboring vertices.

These graphs can be used to extract features. Depending on the selected type of graph, features can take on different values.

3.2 Feature construction

Since only the graph is the source of information, we have used many graph analysis algorithms to construct features. Most algorithms produce results that need to be converted into a numerical metric for the entire graph in order to be used in machine learning as features. Therefore, we construct most of the features in two stages: we apply graph algorithms and then analyze the results by statistical methods.

At first step, we use next graph algorithms [34] that we classify by type of result:

(a) Algorithms that calculate the centrality (some weight) of each vertex in the graph. Thus, we obtain the distribution of the vertex centrality measures. We used several centrality measures: Degree centrality, Closeness centrality, PageRank, VoteRank, Katz centrality, Load centrality, Effective size, Average neighbor degree.

(b) Algorithms that calculate the subset of vertices: Dominating Set, Maximal independent set, Isolates set, Bridges.

(c) Algorithms that calculate the list of subsets of vertices: Cores, Communities detection based on modularity, and Communities detection based on label propagation.

(d) Algorithms that calculate different coefficients: Clustering coefficient, Global efficiency, Degree Assortativity, S-metric, community modularity.

At second step, we calculate next statistics: mean, Q1, Q2, Q3, min, max, standard deviation, number of sets, the relation of a subset to the entire set. The scheme of applying statistical methods to the results of graph algorithms is presented in the Table 1.

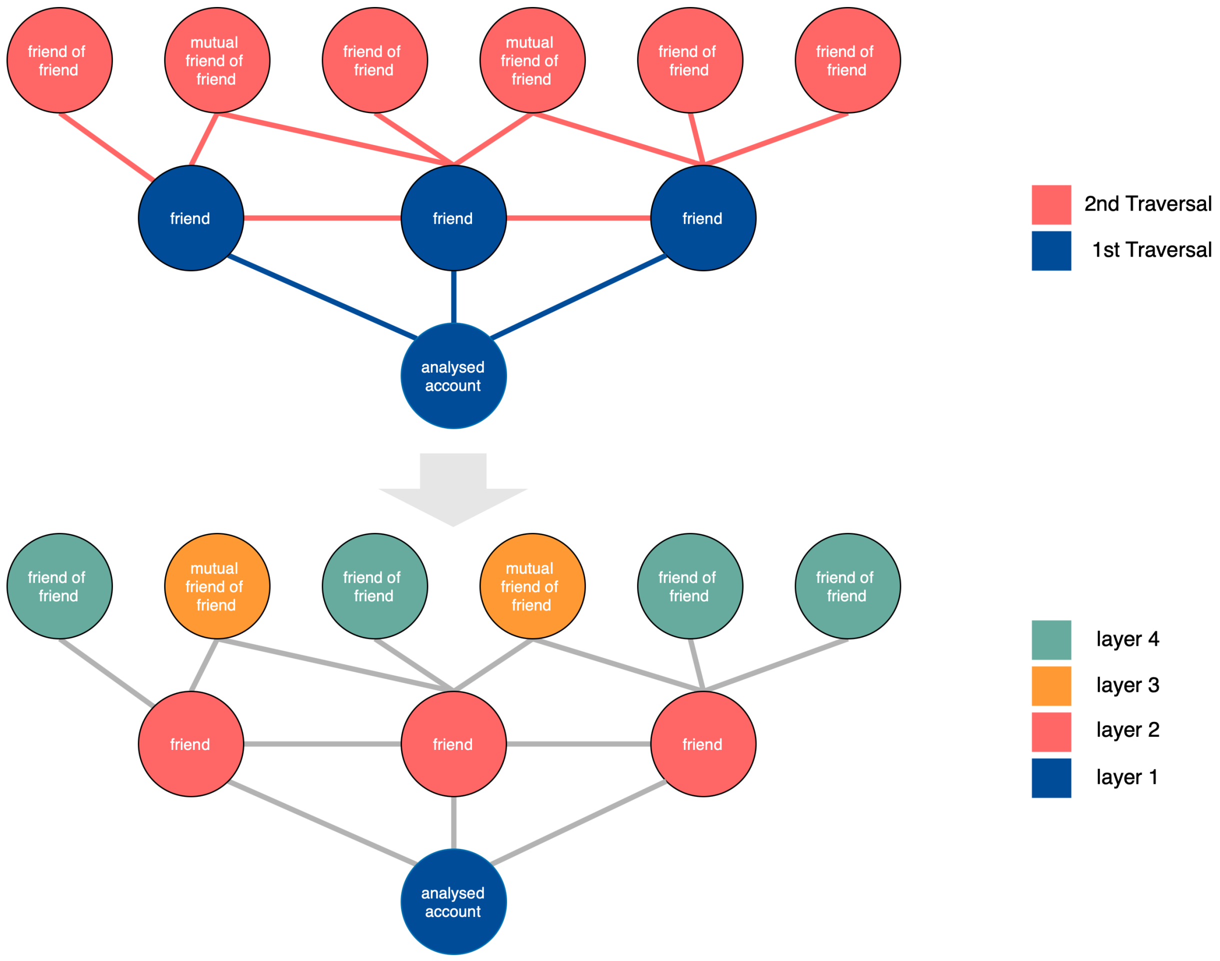


Figure 2: Graph traversal and graph generation scheme.

As a feature, we also use the following data, which we receive along with the graphs: distribution of identifiers (ids) of graph vertices, the id of analyzed account, number of vertices in the graph, number of edges in the graph. As the result, we have 78 features that describe the graph numerically.

3.3 Training the model

Bot detection is a binary classification task: *bot* or *not a bot*. As follows from the reviews [37], the methods that give the best results in bots detection are Random Forest Classifier and Neural Networks. Therefore, we propose using them, together with min-max scaling and grid-search or other hyperparameter optimization.

We propose to evaluate the effectiveness of bot detection using the following metrics:

(a) Area under ROC-curve (AUC-ROC) – the common integrated measure for binary classification tasks. We suggest using it as the main measure of efficiency.

(b) F1 score – a weighted average of the precision and recall that is very popular in papers dedicated to bot detection.

(c) Precision and false-positive rate (FPR) – give an understanding of false positives, which is extremely important for social network protection systems, as it assesses how strong countermea-

step 1		**step 2**	
Graph algorithm	**Result**	**Statistics**	**Features**
Degree	distribution	mean	$8*5=40$ features
Closeness		Q1	
PageRank		Q2	
VoteRank		Q3	
Katz		deviation	
Load			
Effective size			
Average neighborhood degree			
Dominating set	set	$\frac{size(Set)}{size(V)}$	$4*1=4$ features
Maximal independent set			
Isolates set			
Bridges set			
Community (modularity based)	list of sets (distribution of set's sizes)	mean	$2*7=14$ features
		Q1	
		Q2	
		Q3	
Community (label propagation based)		deviation	
		min	
		max	
	modularity	-	2 features
K-cores	list of sets	mean	$1*6=6$ features
		Q1	
		Q2	
		Q3	
		deviation	
		number	
Clustering coefficient	coefficient	-	4 features
Global efficiency			
Degree assortativity			
S-metric			
ids of vertices	-	mean	5 features
		Q1	
		Q2	
		Q3	
		deviation	
id of analyses account	-	-	3 features
number of vertices			
number of edges			

Table 1: Feature construction scheme.

sures can be applied. For example, to automatically block or freeze pages, one need to have an extremely low number of false positives.

(d) Recall – describes the ability to detect bots.

4 Evaluation

To evaluate the effectiveness of the proposed solution, we conducted experiments to detect bots on the VKontakte social network. VKontakte is a predominant Russian social network that is very similar to Facebook. Also VKontakte has an open API.

In this section, we describe the 3 main components of our analysis: (1) obtaining and generating a dataset for analysis and training, (2) feature analysis, and (3) training models with the results of bot detection efficiency.

4.1 Dataset generation

To collect the graphs, we extended the dataset that was collected earlier [24, 27]. This dataset contains a list of bots of different quality and real users from different communities. The bots are collected from three companies that trade bots. For this, fake absurd (so that real users do not join) groups were created, filled with content, and then bought likes from bots in these groups. Bots are divided into 3 qualities, that was evaluated by the description on the website of the company that traded bots:

(a) Low quality (LOW) – bots are probably auto-generated accounts that are controlled by software.

(b) Medium quality (MED) – bots are probably user-like bots that are controlled by software.

(c) High quality (HIGH) – bots are probably controlled by human or users from an exchange platform.

So, the quality of a bot is an expression of how similar a bot is to a real person. We also categorize bots by company, as different bot seller may have different strategies for creating / managing bots:

(a) Company A – exchange platform, where real users are asked to act on a social network for money. One need to set a goal and describe the action to be performed (like, write a negative comment, etc.). The buyer sets the price and sets the required account parameters (gender, age, etc.). Price affects the speed of action - if one set the price higher, then users will respond faster to the offer. We indicated the lowest possible price and did not include requirements for parameters.

(b) Company B – an online store of bots, where one need to specify the goal, the required action and the quality of the bots.

(c) Company C – similar online store of bots.

Users are real social network random users collected from separate social network groups.

A summary of bots and user data is given in the Table 2.

In order to implement data collection, we used the API VKontakte with the VK-API python wrapper. VKontakte allows one to combine up to 25 API requests in 1 request. Therefore, the API complexity of obtaining data in our case was 25 times lower.

Knowing the id of bots and users, we have collected graphs of friends, excluding analyzed account, mutual friends of friends, and not mutual friends of friends (layers 1, 3, and 4) – as shown in Figure 3. A graph of this type was further used in experiments.

A dataset with the graphs of bots and real users we have collected is available via the link [24].

Dataset	Type	Company	Count	Descriptions
bot_1	LOW	C	301	Probably bots that are controlled by software.
bot_2	MED	C	295	Probably more user-like bots that are controlled by software.
bot_3	HIGH	C	298	Probably bots that are controlled by human or users from exchange platform.
bot_4	LOW	B	301	Probably bots that are controlled by software.
bot_5	MED	B	303	Probably more user-like bots that are controlled by software.
bot_6	HIGH	B	304	Probably bots that are controlled by human or users from exchange platform.
bot_7	HIGH	A	302	Probably users from the exchange platform.
bot_8	HIGH	A	357	Probably users from the exchange platform with more filled profiles.
user_1	activists community		385	Users from group ”velosipedization” that is dedicated to development of bicycle transport.
user_2	mass media community		298	Users from group ”belteanews” that is dedicated to Belarussian news.
user_3	developers community		332	Users from group ”tproger” that is dedicated to software development.
user_4	sport community		224	Users from group ”mhl” that is dedicated to youth hockey.
user_5	mass media community		420	Users from group ”true_lentach” that is dedicated to Russian news.
user_6	blogger community		251	Users from group ”mcelroy_dub” that is dedicated to re-playing of funny videos.
user_7	commerce community		284	Users from group ”sevcableport” that is dedicated to creative space.
user_8	festival community		259	Users from group ”bigfestava” that is dedicated to cartoon festival.
user_9	sport community		181	Users from group ”hcakbars” that is dedicated to fun community of hockey club.
user_10	developers community		397	Users from group ”tnull” that is dedicated to software development and memes.

Table 2: Description of a dataset for experiments, which includes bots of different quality from different companies as well as real users.

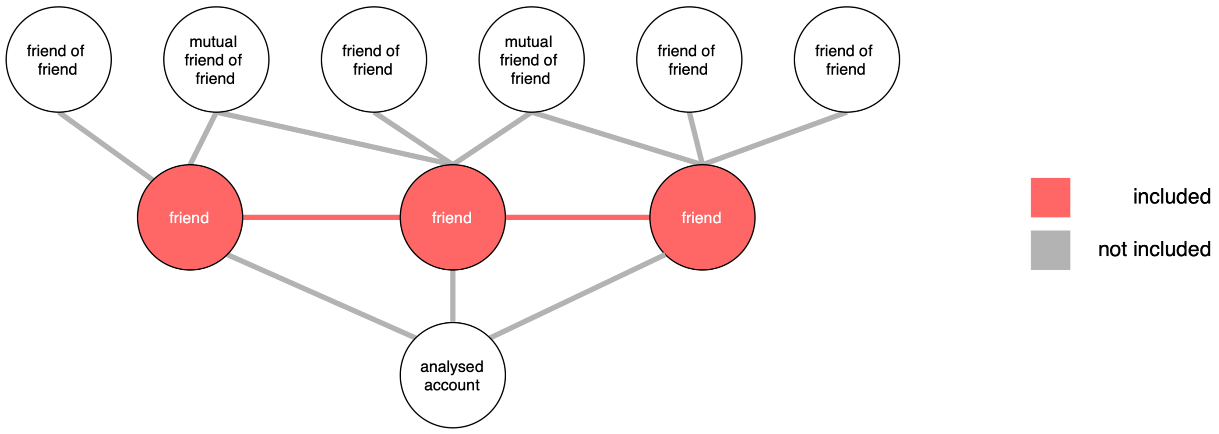


Figure 3: Graph generation for experiments – we leave only the friends of the bot and the connections between them.

4.2 Feature analysis

To construct features, we used Python and networkX library. We perform feature analysis by two parameters:

(a) The presence of a very high correlation between features. The presence of a very high correlation between features indicates that they are duplicates, and one of the features can be removed - which will not significantly affect the machine learning outcomes. This can be used to reduce the number of feature calculations, thereby increasing performance.

(b) The presence of a correlation between the features and the label, which indicates that the correlated feature allows one to explicitly divide the dataset. This analysis provides a general understanding of the apparent usefulness of the features.

After a round of analyzing the correlation between features, we removed features that correlated greater than 0.9. The resulting features that will be used in training are presented in the form of a correlation matrix in Figure 4. A table with the correlation of feature and label is shown in Figure 5. Higher values indicate that the feature is more capable of linearly partitioning the dataset.

4.3 Evaluation results

In this paper, we used a dataset in which bots are divided by the quality and by company. This allows us to assess how resistant the proposed bot detection approach is to new types of bots. That's why we propose efficiency tests according to 4 schemes:

(a) Standard test. Estimates in Table 3 are given for tests in which the division into training and validation samples was random across the entire dataset. This means that the training and validation set included bots of all companies and of all qualities.

(b) Standard test with feature selection. Estimates in Table 4 are given for standard tests in which features with very high correlation were removed. This test will allow one to determine whether the dimension decreasing of the feature set affected the quality of the classification.

(c) Bot quality resistance test (cross quality LOW/MED/HIGH). In Table 5 the training sample contains bots of one quality, and the validation sample contains another. Such a test shows the resistance of the method to the quality of bots. Both samples also include users.

(d) Test of resistance to bot management strategy (cross company A/B/C). In Table 6 the training sample contains bots of one company, and the validation sample contains another. Such a test shows the resistance of the method to the quality of bots to management strategies that differs from one bot company to another. Both samples also include users.

We conduct the experiments using two models: a random forest and a neural network. For implementation, we used the Keras and scikit-learn libraries. All tests solve the problem of binary classification (bot or not bot).

For random forest, we performed a simple grid search hyperparameter optimization. In the neural network, we used one hidden layer, and the output layer was a single neuron with a sigmoid activation function. We used optimization for the hidden layer size and the number of epochs. In all tests, we also used min-max scaling.

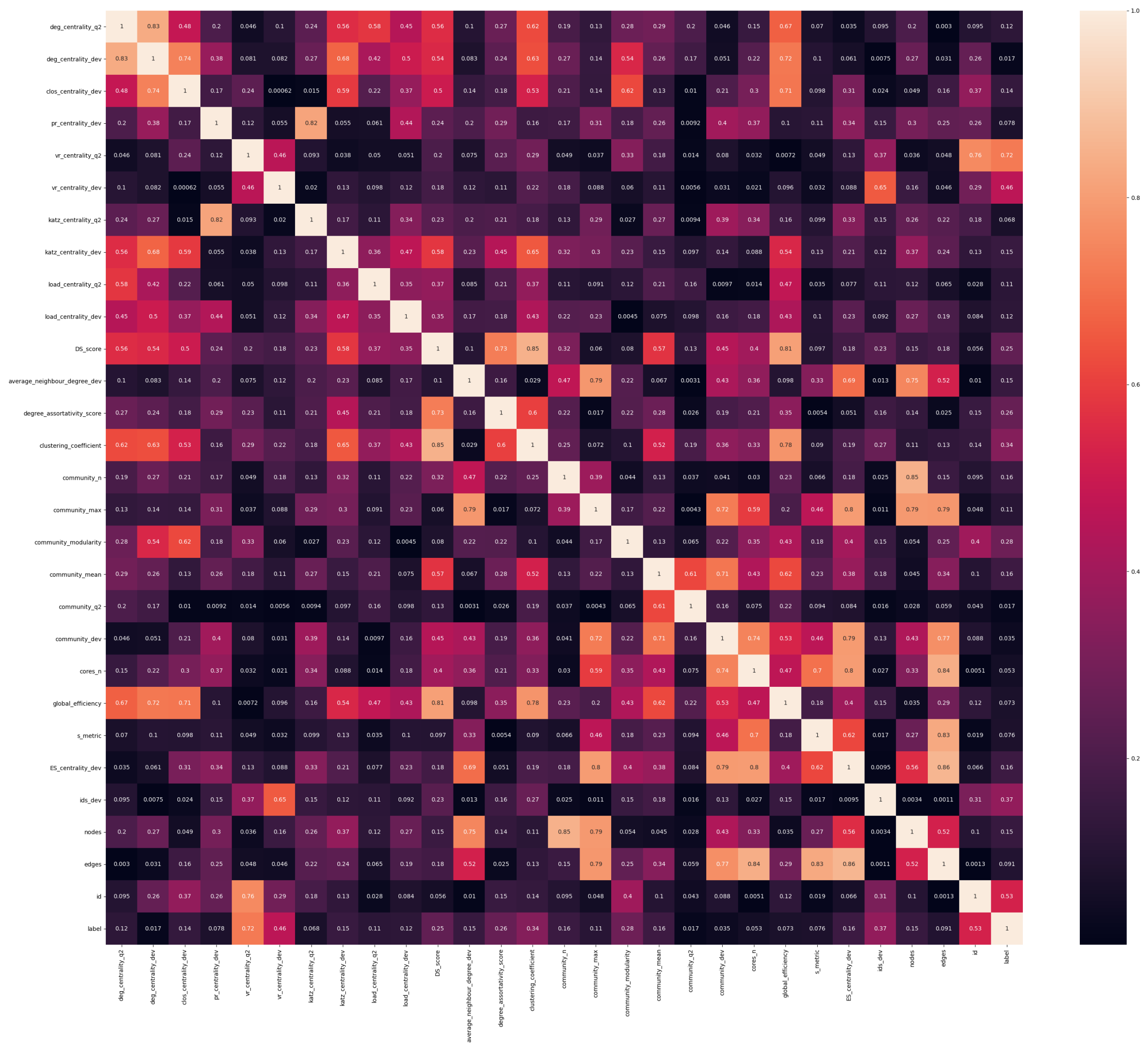


Figure 4: Modulo correlation between features that are lower 0.9.

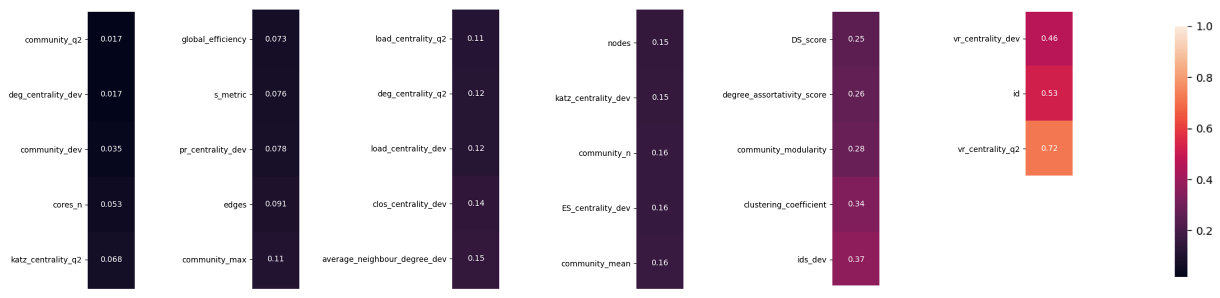


Figure 5: Modulo correlation between features and label.

Model	Accuracy	FPR	Precision	Recall	F1	AUC-ROC
RFC	0.891	0.070	0.904	0.843	0.872	0.886
NN	0.885	0.070	0.903	0.828	0.855	0.938

Table 3: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in standard test.

Model	Accuracy	FPR	Precision	Recall	F1	AUC-ROC
RFC	0.896	0.062	0.915	0.845	0.878	0.891
NN	0.884	0.088	0.849	0.849	0.857	0.939

Table 4: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in the standard test with feature selection, where features with very high correlation were removed.

Model and split strategy	Accuracy	FPR	Precision	Recall	F1	AUC-ROC
RFC train=MED, HIGH test=LOW	0.834	0.170	0.711	0.842	0.771	0.836
RFC train=LOW, HIGH test=MED	0.905	0.122	0.813	0.953	0.878	0.915
RFC train=LOW, MED test=HIGH	0.629	0.048	0.903	0.366	0.521	0.659
Summary $\bar{x} \pm \sigma$	0.789 ± 0.143	0.113 ± 0.061	0.809 ± 0.096	0.720 ± 0.312	0.723 ± 0.183	0.803 ± 0.131
NN train=MED, HIGH test=LOW	0.829	0.218	0.678	0.922	0.328	0.933
NN train=LOW, HIGH test=MED	0.872	0.175	0.753	0.956	0.364	0.954
NN train=LOW, MED test=HIGH	0.715	0.089	0.883	0.554	0.370	0.835
Summary $\bar{x} \pm \sigma$	0.805 ± 0.081	0.161 ± 0.066	0.771 ± 0.104	0.810 ± 0.223	0.354 ± 0.023	0.907 ± 0.064

Table 5: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in the quality-split test, where the model was not trained on the data of one of the qualities and is trying to recognize it.

5 Discussion

The experimental results show that using the proposed approach, it is possible to achieve high detection quality. The main quality metric is the area under ROC-curve (AUC-ROC), which is a common metric for binary classification tasks. We also pay special attention to the metrics characterizing false positives - Precision with the false-positive rate (FPR).

These results are comparable to those obtained by other researchers. As follows from the reviews [37, 20], most of the classifiers developed by researchers have about 0.9 AUC ROC or higher. Table 3

Model and split strategy	Accuracy	FPR	Precision	Recall	F1	AUC-ROC
RFC train=A, B test=C	0.787	0.046	0.851	0.483	0.616	0.718
RFC train=A, C test=B	0.782	0.076	0.878	0.621	0.727	0.773
RFC train=C, B test=A	0.918	0.093	0.895	0.931	0.912	0.919
Summary $\bar{x} \pm \sigma$	0.829 ± 0.077	0.072 ± 0.024	0.875 ± 0.022	0.678 ± 0.229	0.752 ± 0.150	0.803 ± 0.104
NN train=A, B test=C	0.725	0.161	0.637	0.516	0.240	0.636
NN train=A, C test=B	0.750	0.147	0.790	0.631	0.338	0.817
NN train=C, B test=A	0.851	0.259	0.763	0.982	0.461	0.971
Summary $\bar{x} \pm \sigma$	0.775 ± 0.067	0.189 ± 0.061	0.730 ± 0.082	0.710 ± 0.243	0.346 ± 0.111	0.808 ± 0.168

Table 6: Classification performance for Random Forest Classifier (RFC) and Neural Network (NN) in the company-split test, where the model was not trained on the data of one of the companies and is trying to recognize it.

shows that it is possible to achieve very good performance (AUC ROC more than 0.9 and FPR lower than 0.1 for neural network) relying only on data about the friends graph. Moreover, the Table 4 shows that the number of features can be significantly reduced, and deleting a number of highly correlated features doesn't affect much the classification quality.

Tables 5 and 6 show that it is extremely important to have bots of different quality and from different companies in the training sample. Not all types of bots in the training sample are interchangeable. We assess these results as the ability of our approach to recognizing if bots mutate (change their control strategy [44]). But at the same time, the average AUC-ROC is still ≈0.9 for quality and ≈0.8 for company tests.

More detailed conclusions can be drawn if one look at the components of these tables.

From cross quality test (Table 5) we can draw a slightly trivial conclusion that by training on high-quality bots (table rows with *train=HIGH*), it will be possible to recognize low and medium-quality bots with greater efficiency (AUC-ROC ≈ 0.9) in comparison with training only on low and medium-quality bots (AUC-ROC ≈ 0.8). This refers to the fact that it is better to include high quality bots in training datasets.

Cross company test (Table 6) shows that the quality of the classification is more influenced by the management strategy implemented by the company. For example, one company (compared to another) may pay more attention to creating a good friend structure for a bot. Therefore, the estimates for split by companies vary greatly.

An important aspect of the proposed approach is also that it is only based on the analysis of a graph formed by friends. Such technique has several advantages:

- A small amount of information that is needed for analysis. Unlike text analysis, statistical data and media content, graph analysis does not require a lot of information to download. The amount of information required can be expressed in API requests, the number of which is equal to the number of user's friends (*API complexity = number of friends + 1*).

- There are no language features. Language features depend on many parameters, which are often difficult to take into account together. The texts can be very different for people who speak different languages, from different countries, with different levels of income, education, age, etc. Graph structures are a more universal source of information and do not depend on the characteristics of various social groups.

- If the account is closed by the privacy settings, the list of its friends can be set indirectly. To do this, one needs to get a graph of friends of the entire social network and find the analyzed account in the friendslists of other users. Getting a graph of friends for the entire social network is a difficult, but a completely solvable task. This means that the graph-based approach allows one to detect bots whose profiles are closed by privacy settings or even blocked.

- It is more difficult for bots to mimic the natural structure of a friend graph (in comparison with filling the profile and content). The user's friend graph consists of many overlapping communities and his friends are highly connected. The bot is forced to add random people (who most likely will not be connected) as friends. Or the bot will try to mimic the structure of friends, which will have anomalous features.

Globally - if think about bot detection task as a part of countering information attacks on social networks, we believe that the effectiveness of the proposed approach allows it to be used with a number of countermeasures. Social networks can use it to detect bots and apply soft severity countermeasures (captcha) and medium severity countermeasures (speed reduction, verification requirement, etc.). The small amount of data required for analysis also allows the use of an approach not only for social networks but also for administrators of online communities to clear participants from bots.

The aspect of applicability should also be noted. The presented experiment was conducted on the social network VKontakte. Despite the fact that there are graphs of friends in almost all social networks, the effectiveness of the proposed solutions for other platforms should be checked separately. We believe that social networks that are similar to VKontakte will have similar results, because their users form similar structures - for example, Facebook and LinkedIn.

6 Conclusion

Bot detection is an increasingly relevant feature in the social media security arsenal. Therefore, researchers are actively looking for new detection models and data sources for bot detection.

We have developed an approach for bot detection that relies on only one data source - the bot's friends graph. Experiments have shown that based on graphs alone, it is possible to achieve high detection quality (AUC-ROC=0.938) with a sufficiently low number of false positives (Precision=0.903) to apply automatic countermeasures of medium severity.

Unlike other approaches, which are based on the analysis of text, bot profile and generated content, our solution has several advantages: language independence, a small number of requests, and the ability to analyze hidden accounts.

We have demonstrated that the approach is sufficiently robust to recognize mutating bots - bots of a new quality (AUC-ROC$\approx$0.9) and from new companies (AUC-ROC$\approx$0.8).

We experimented with bots from the social network VKontakte and open access to our dataset [24]. The proposed approach can be applied to all social networks that have user graphs, but we believe that the experimental results can be extrapolated only to social networks similar to VKontakte - for example, Facebook and LinkedIn.

Future Work. In the future, we plan to significantly enhance the proposed approach through new data sources and feature construction methods. We will try to use several graph structures at once (graphs of mutual friends, graphs of commentators, etc.), as well as statistical information about adjacent accounts. We will also try to construct features based on graph embedding methods.

Acknowledgments

This research was supported by the Russian Science Foundation under grant number 18-71-10094 in SPC RAS.

References

[1] K. S. Adewole, N. B. Anuar, A. Kamsin, K. D. Varathan, and S. A. Razak. Malicious accounts: Dark of the social networks. *Journal of Network and Computer Applications*, 79:41–67, February 2017.

[2] S. Al-Azani and E.-S. M. El-Alfy. Detection of arabic spam tweets using word embedding and machine learning. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT'18), Sakhier, Bahrain*, pages 1–5. IEEE, November 2018.

[3] S. M. B. K. S. Asis and K. Chattopadhyay. *Statistical methods in social science research*. Springer, 2018.

[4] U. author. search4faces – service for finding people by photo. `https://search4faces.com/vk01/index.html` [Online; accessed on June 22, 2021], 2021.

[5] N. Chavoshi, H. Hamooni, and A. Mueen. Debot: Twitter bot detection via warped correlation. In *Proc. of the IEEE 16th International Conference on Data Mining serie (ICBM'16), Barcelona, Spain*, pages 817–822, December 2016.

[6] C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min. Statistical features-based real-time detection of drifted twitter spam. *IEEE Transactions on Information Forensics and Security*, 12(4):914–925, October 2016.

[7] C. A. Davis, O. Varol, E. Ferrara, A. Flammini, and F. Menczer. Botornot: A system to evaluate social bots. In *Proc. of the 25th international conference companion on world wide web (WWW'16), Québec, Montréal, Canadaa*, pages 273–274. ACM, April 2016.

[8] J. P. Dickerson, V. Kagan, and V. Subrahmanian. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *Pro.of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'14), Beijing, China*, pages 620–627. IEEE, August 2014.

[9] G. Dong and H. Liu. *Feature engineering for machine learning and data analytics*. CRC Press, march 2018.

[10] A. Dorri, M. Abadi, and M. Dadfarnia. Socialbothunter: Botnet detection in twitter-like social networking services using semi-supervised collective classification. In *Proc. of the 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/-CyberSciTech'18), Athens, Greece*, pages 496–503. IEEE, August 2018.

[11] V. Gómez, A. Kaltenbrunner, and V. López. Statistical analysis of the social network and discussion threads in slashdot. In *Proc. of the 17th international conference on World Wide Web (WWW'08), Beijing, China*, pages 645–654, April 2008.

[12] Google. Vision ai. `https://cloud.google.com/vision` [Online; accessed on June 22, 2021], 2021.

[13] C. Grimme, M. Preuss, L. Adam, and H. Trautmann. Social bots: Human-like by means of human control? *Big data*, 5(4):279–293, December 2017.

[14] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proc. of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'16), California, San Francisco, USA*, pages 855–864. ACM, August 2016.

[15] M. Heidari and J. H. Jones. Using bert to extract topic-independent sentiment features for social media bot detection. In *Proc. of the 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON'20), New York, New York, USA*, pages 0542–0547. IEEE, October 2020.

[16] M. Heidari, J. H. Jones, and O. Uzuner. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. In *Proc. of the 2020 International Conference on Data Mining Workshops (ICDMW'20), Sorrento, Italy*, pages 480–487. IEEE, November 2020.

[17] C.-C. Hsu, C.-Y. Lee, and Y.-X. Zhuang. Learning to detect fake face images in the wild. In *Proc. of the 2018 International Symposium on Computer, Consumer and Control (IS3C'18), Taichung, Taiwan*, pages 388–391. IEEE, December 2018.

[18] Z. Jin, J. Cao, Y. Zhang, J. Zhou, and Q. Tian. Novel visual and statistical image features for microblogs news verification. *IEEE transactions on multimedia*, 19(3):598–608, October 2016.

[19] A. Kamal and M. Abulaish. Statistical features identification for sentiment analysis using machine learning techniques. In *Proc. of the 2013 International Symposium on Computational and Business Intelligence (ISCBI'13), New Delhi, India*, pages 178–181. IEEE, August 2013.

[20] A. Karataş and S. Şahin. A review on social bot detection techniques and research directions. In *Proc. Int. Security and Cryptology Conference Turkey*, pages 156–161, October 2017.

[21] I. Karpov and E. Glazkova. Detecting automatically managed accounts in online social networks: Graph embedding approach. *arXiv preprint arXiv:2010.07923*, 1357:11–21, October 2020.

[22] V. Këpuska and G. Bohouta. Comparing speech recognition systems (microsoft api, google api and cmu sphinx). *Int. J. Eng. Res. Appl*, 7(03):20–24, March 2017.

[23] M. Kolomeec, A. Chechulin, and I. Kotenko. Methodological primitives for phased construction of data visualization models. *Journal of Internet Services and Information Security*, 5(4):60–84, 2015.

[24] M. Kolomeets. Security datasets. `https://github.com/guardeec/datasets` [Online; accessed on June 22, 2021], February 2021.

[25] M. Kolomeets, A. Benachour, D. El Baz, A. Chechulin, M. Strecker, and I. Kotenko. Reference architecture for social networks graph analysis tool. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 10(4):109–125, December 2019.

[26] M. Kolomeets, A. Chechulin, and I. V. Kotenko. Social networks analysis by graph algorithms on the example of the vkontakte social network. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 10(2):55–75, June 2019.

[27] M. Kolomeets, O. Tushkanova, D. Levshun, and A. Chechulin. Camouflaged bot detection using the friend list. In *Proc. of the 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'21), Valladolid, Spain*, pages 253–259. IEEE, March 2021.

[28] I. V. Kotenko, I. Saenko, and A. Kushnerevich. Parallel big data processing system for security monitoring in internet of things networks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 8(4):60–74, December 2017.

[29] M. Kuptsov, V. Minaev, S. Yablochnikov, V. Dzobelova, and A. Sharopatova. Some statistical features of the information exchange in social networks. In *Proc. of the 2020 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO'20), Svetlogorsk, Russia*, pages 1–4. IEEE, July 2020.

[30] H. Mansourifar and W. Shi. One-shot gan generated fake face detection. *arXiv preprint arXiv:2003.12244*, March 2020.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, January 2013.

[32] A. Minnich, N. Chavoshi, D. Koutra, and A. Mueen. Botwalk: Efficient adaptive exploration of twitter bot networks. In *Proc. of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM'17), Sydney, Australia*, pages 467–474, July 2017.

[33] L. Nataraj, T. M. Mohammed, B. Manjunath, S. Chandrasekaran, A. Flenner, J. H. Bappy, and A. K.

Roy-Chowdhury. Detecting gan generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019(5):532–1, January 2019.

[34] M. Newman. *Networks*. Oxford university press, July 2018.

[35] M. E. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *Proc. of the national academy of sciences*, 99(suppl 1):2566–2572, February 2002.

[36] B. Oberer, A. Erkollar, and A. Stein. Social bots–act like a human, think like a bot. In *Digitalisierung und Kommunikation*, volume 31 of *EKW*, pages 311–327. Springer, April 2019.

[37] M. Orabi, D. Mouheb, Z. Al Aghbari, and I. Kamel. Detection of bots in social media: a systematic review. *Information Processing & Management*, 57(4):102250, July 2020.

[38] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proc. of the 2014 conference on empirical methods in natural language processing (EMNLP'14), Doha, Qatar*, pages 1532–1543, October 2014.

[39] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proc. of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'14), New York, New York, USA*, pages 701–710. ACM, August 2014.

[40] M. Polignano, M. G. de Pinto, P. Lops, and G. Semeraro. Identification of bot accounts in twitter using 2d cnns on user-generated contents. In *Proc. of the CLEF (Working Notes)*, September 2019.

[41] G. Saidov and Y. Zdanovich. Find clone – service for finding people by photo. `https://findclone.ru` [Online; accessed on June 20, 2021], 2021.

[42] P. Shi, Z. Zhang, and K.-K. R. Choo. Detecting malicious social bots based on clickstream sequences. *IEEE Access*, 7:28855–28862, February 2019.

[43] K. Skorniakov, D. Turdakov, and A. Zhabotinsky. Make social networks clean again: Graph embedding and stacking classifiers for bot detection. In *Proc. of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18), Turin, Italy*. ACM, October 2018.

[44] T. Stein, E. Chen, and K. Mangla. Facebook immune system. In *Proc. of the 4th workshop on social network systems (SNS'11), Salzburg, Austria*, pages 1–8, April 2011.

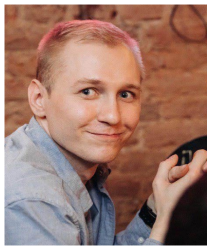
[45] V. S. Subrahmanian, A. Azaria, S. Durst, V. Kagan, A. Galstyan, K. Lerman, L. Zhu, E. Ferrara, A. Flammini, and F. Menczer. The darpa twitter bot challenge. *Computer*, 49(6):38–46, June 2016.

[46] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, November 2011.

[47] P. Wang. thispersondoesnotexist – service for generating photos of non-existent people. `https://thispersondoesnotexist.com` [Online; accessed on June 22, 2021], 2021.

[48] F. Wei and U. T. Nguyen. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In *Proc. of the 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA'19), Los Angeles, California, USA*, pages 101–109. IEEE, December 2019.

Author Biography



Maxim Kolomeets is currently a PhD student under joint supervision of ITMO University and University Toulouse 3 Paul Sabatier. He is junior researcher at the Laboratory of Computer Security Problems of St.Petersburg Federal research center of the Russian Academy of Sciences (SPC RAS). His research interests include social network analysis, distributed system security, and security visualization. He is the author of more than 20 refereed publications.

Andrey Chechulin received his B.S. and M.S. in Computer science and computer facilities from Saint-Petersburg State Polytechnical University and PhD from St.Petersburg Federal research center of the Russian Academy of Sciences (SPC RAS) in 2013. In 2015 he was awarded the medal of the Russian Academy of Science in area of computer science, computer engineering and automation. At the moment he holds a position of leading researcher at the Laboratory of Computer Security Problems of SPC RAS. He is the author of more than 80 refereed publications and has a high experience in the research on computer network security and participated as an investigator in several projects on developing new security technologies. His primary research interests include computer network security, intrusion detection, analysis of the network traffic and social network analysis.

Igor Kotenko graduated with honors from St.Petersburg Academy of Space Engineering and St. Petersburg Signal Academy. He obtained the Ph.D. degree in 1990 and the National degree of Doctor of Engineering Science in 1999. He is Professor of computer science and Head of the Laboratory of Computer Security Problems of St.Petersburg Federal research center of the Russian Academy of Sciences (SPC RAS). He is the author of more than 500 refereed publications, including 14 textbooks and monographs. Igor Kotenko has a high experience in the research on computer network security and participated in several projects on developing new security technologies. For example, he was a project leader in the research projects from the US Air Force research department, via its EOARD (European Office of Aerospace Research and Development) branch, EU FP7 and FP6 Projects, HP, Intel, F-Secure, etc. The research results of Igor Kotenko were tested and implemented in more than fifty Russian research and development projects.