Detection of Social Bots in Social Network “VKontakte”

Filipp Beldiushkin
(Student, Yuri Shichalin’s Classical Lyceum)

Abstract

This report demonstrates how to identify bots in social media. The study focuses on a Russian social network called “VKontakte”. This research was conducted using classical machine learning methods and graphical approaches. All the information was taken from a user’s profile page and indicated network of friends on a profile page.

1 Introduction

Social bots are used to spread spam messages and fake news. Fake news are often introduced in the period of political elections to alternate the course of election or influence financial markets. The reason why social media is the best source for spreading fake news is because a user of a social platform will be more likely to believe information that comes from a trusted “circle of friends”, instead of “government-controlled”, less familiar media outlets.

Detecting bot profiles is one of the pressing tasks of modern IT technologies, as it helps to reduce the theft of users' personal data, halt the spread of fake news, increase the objectivity of social research, voting, etc. At the moment, there are two main methods of identifying fake accounts: through the study of personal data and through the detection of abnormal activity. Both of these methods are based on numerical statistical analysis of the user's profile and activity. However, both of these methods do not take into account the user's social connections.

This report discusses one of the ways to detect bots in the social network VKontakte. More specifically, the research was done by analyzing connections a user has, which are generally called “friends”. The main hypothesis of this research is the following: social bots choose friends from random users, while real users choose friends from their social circle.

2 Data Collection

To collect data from a user’s profile, open APIs of VKontakte social network were used.

1,797 IDs of social bots were downloaded from the website gosvon.net . 9,969 IDs of real users were selected randomly from VKontakte.

Data of two types was collected for each account:

1. Data from a user’s profile (listed in Appendix)
2. Graph features extracted from the network of friends

The main attention in this report is paid to the graph of friend connections created from a user’s profile information. The graph was built based on the following assumptions: for each of the 11,766 accounts (denoted as “Set V1”), their friends were taken (denoted as “Set V2”). For each of the accounts of the V2 set, their friends were taken (denoted as “Set V3”). After that, for each account from the set V1, a graph Gv was constructed , which consisted of the node v. Its adjacent nodes were taken from the set V2. Node v is connected to all the nodes from the set V2, and all the nodes from V2 are also connected to those in V2 and V3. The edges of this graph represent friend connections. Due to the observed symmetry of these friend connections, the edges of the graph are undirected, and the graph itself is undirected.

Then, the graph features were calculated for each of the Gvs. The features listed in Appendix quantitatively describe the structure of a user's friend network. There are two examples shown in Figure 1 and Figure 2.

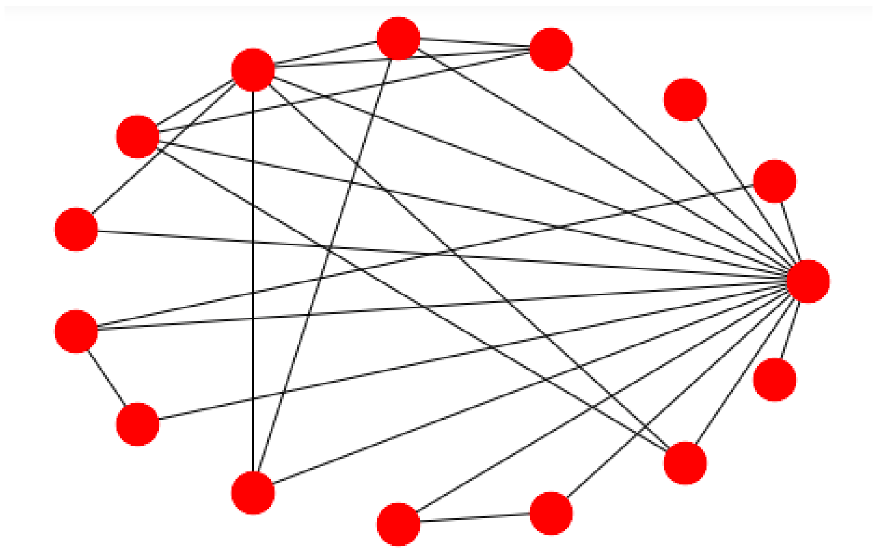

Figure 1: Graph G_v visualizes data collected from the profile of a real user

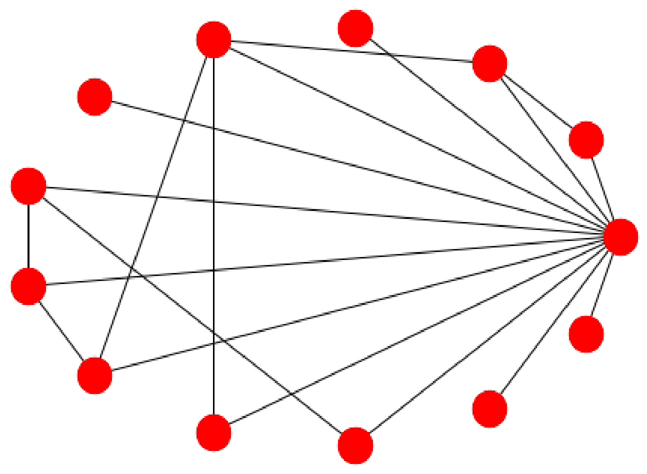

Figure 2: Graph G_v sualizes data collected from the profile of a social bot

3 Approach

The main hypothesis of this research is the following: to determine whether or not a user of Vkontakte network is a social bot, one should examine the friend network of this particular user. The efficiency of using graph features has to be compared to the efficiency of profile features.

The algorithm for this comparison is as follows:

1. Classification model is built;
2. The significance of each feature is measured using Permutation Importance algorithm;
3. The rank of the significance of listed graphical features is recorded.

Three methods were used to build classification models:

1. Logistic Regression;
2. Random Forest;
3. Gradient Boosting.

Permutation Importance algorithm includes the following steps :

1. A validation dataset is fixed;
2. The quality of the model is calculated on the validation dataset;
3. One feature is removed from the dataset forming a revised dataset;
4. The quality of the model is calculated on the revised dataset;
5. The difference between the qualities is calculated.

Thus, the Permutation Importance algorithm demonstrates that the feature is important if its removal leads to the significant drop in quality.

4 Experimental Data and Results

Random Forest algorithm showed the most statistically relevant result (Table 1).

Method of modelling	**roc_auc score**
Logistic Regression	0.876
Random Forest	0.981
Gradient Boosting	0.975

Table 1: Results of the experiment after running the Random Forest algorithm.

Top five features according to Permutation Importance in Logistic Regression (Table 2).

Weight	**Feature**
0.0701 ± 0.0053	average_neighbor_degree
0.0289 ± 0.0017	closeness_centrality
0.0419 ± 0.0034	subscriptions
0.0178 ± 0.0013	relation
0.0166 ± 0.0012	transitivity

Table 2

Top five features according to Permutation Importance in Random Forest (Table 3).

Weight	Feature
0.2950 ± 0.0062	average_neighbor_degree
0.0747 ± 0.0025	pages
0.0419 ± 0.0034	has_photo
0.0101 ± 0.0009	subscriptions
0.0088 ± 0.0003	friends

Table 3

Top five features according to Permutation Importance in Gradient Boosting (Table 4).

Weight	Feature
0.2849 ± 0.0055	average_neighbor_degree
0.1399 ± 0.0033	has_photo
0.1147 ± 0.0012	pages
0.0871 ± 0.0039	closeness_centrality
0.0158 ± 0.0012	friends

Table 4

5 Conclusion

The results of the experiment show that the graph analysis (namely average_neigbor_degree feature) gives the best indication on how to distinguish a social bot from a real user. This confirms the original hypothesis that social bots choose friends from random users, while real users choose friends from their social circle, as the average_neighbor_degree feature indicates how many friends of the studied users are actually friends with each other.

Finally, it is important to emphasize that social bots can be successfully detected with the help of the built models. As it was demonstrated in this experiment, fabricating a network of friends is much more complicated than fabricating fake profile data.

References and Acknowledgements

- Alymov A.S., Baranyuk V.V., Smirnova O. S. (2016) Detection of bot programs that mimic human behavior on the VKontakte social network. International Journal of Open Information Technologies, 8: 55–60.
- Boshmaf Y., Muslukhov I., Beznosov K., and Ripeanu M. (2011) The Socialbot Network: When Bots Socialize for Fame and Money. Proceedings of the 27th Annual Computer Security Applications Conference: 93–102 .
- Onur Varol, Emilio Ferrara, Clayton A.Davis, Filippo Menczer, Alessandro Flammini, “Online Human-Bot Interactions: Detection, Estimation, and Characterization”. arXiv:1703.03107v2 [cs.SI]. Mar 27, 2017.
- Bradshaw, Samantha and Philip N. Howard (2019) The Global Disinformation Disorder: 2019 Inventory of Social Media Manipulation. Computational Propaganda Project Working Paper Series. Working Paper 2019.3.

Appendix

The list of the features used in modelling and their description is presented below.

Profile features:

1. has_photo – information about whether the user has a profile photo. Return values: 1 — has, 0 — does not have.
2. sex - user's gender . Return values: 1 —female, 2 —male, 0 — not indicated.
3. has_mobile – information whether a user has a mobile app associated with this social network. Return values: 1 — has, 0 —does not have.
4. followers_count - number of followers a user has.
5. contacts – indication of the phone number on a profile page. Return values: 1 —indicated, 0 — not indicated.
6. relatives - indication of the list of relatives on a profile page. Return values: 1 —indicated, 0 — not indicated.
7. relation – indication of whether a user is in a relationship, and if his/her status is indicated on a profile page. . Return values: 1 —indicated, 0 — not indicated.
8. personal – indication of a relationship status on a profile page. Return values: 1 —indicated, 0 — not indicated.
9. activities – indication of any activities on a profile page. Return values: 1 —indicated, 0 — not indicated.
10. music – indication of favorite music on a profile page. Return values: 1 —indicated, 0 — not indicated.
11. movies – indication of any favorite movies on a profile page. Return values: 1 —indicated, 0 — not indicated.
12. tv – indication of any favorite tv shows on a profile page. Return values: 1 —indicated, 0 — not indicated.
13. books – indication of any favorite books on a profile page. Return values: 1 —indicated, 0 — not indicated.
14. about – indication whether 'about myself' field on profile was filled. Return values: 1 — filled, 0 — not filled.
15. quotes – indication of any favorite quotes on a profile page. Return values: 1 —indicated, 0 — not indicated.
16. albums – number of photo albums.
17. audios – number of audios.
18. followers – number of followers.
19. friends – number of friends.
20. pages – number of pages a user flagged as "interesting".
21. photos – number of photos.
22. subscriptions – number of subscriptions.
23. videos – number of videos.

24. age – age.
25. city – geographical location of a user (city) .
26. country – geographical location of a user (country) .

Graph features:

1. avg_cl - average clustering of the graph, based on a user's network of friends. It is the average of the sum of the results of division of all existing links between the neighbors of an edge and all possible links between them for all nodes in the social graph of a user.
2. trans - transitivity of the graph. It is the relation of the tripled number of triangles to the number of triples of nodes in the social graph of the user.
3. deg_centr - average of the degree centrality, which is the relation of the degree of the node to the number of nodes in the graph minus 1, for every node in the social graph of a user.
4. average_neighbor_degree - average of the average neighbor degree for every node in the social graph of a user.
5. average_degree_connectivity – average of the average nearest neighbor degree, which is the relation of the sum of the degrees of the neighbors of a node to the degree of it, for all nodes in the social graph of a user.
6. k_nearest_neighbors - average of the average neighbor degree for every node in the social graph of a user.
7. degree_centrality – average of the degree centrality, which is the relation of the degree of the node to the number of nodes in the graph minus 1, for every node in the social graph of a user.
8. closeness_centrality – the average of the closeness centrality, which is the reciprocal of the sum of the length of the shortest paths between the node and all other nodes, for every node in the social graph of a user.
9. betweenness_centrality – average of the betweenness centrality, which is the sum of the relations of the number of shortest path length for all combinations of 2 nodes that pass through the given node v to the number of shortest path length for all combinations of 2 nodes, for every in the social graph of a user.
10. diameter - the longest path between any points in the social graph of a user.
11. average_shortest_path_length – average of the lengths of the shortest path lengths between all pairs of nodes in the graph.