Research and analysis of links in social networks

M.I. Khotilin[1], A.V. Blagov[1], I.A. Rytsarev[1]

[1]*Samara National Research University, Moskovskoye shosse, 34, 443086, Samara, Russia*

Abstract

This paper is devoted to the analysis of data and links in social networks. The approach of representation of a social network in the form of a graph is considered. The algorithms for finding communities and main nodes ("hubs"), which are the accounts that have the greatest impact on communities, have been explored and planned for finalization. Existing software environments for visualizing social network data are explored, a software package is developed.

Keywords: social networks; big data; graph; adjacency matrix; SCAN-algorithm; Gephi

1. Introduction

Over the past decade, social networks have played a huge role in the life of society. They, being the subject of socialization of people, occupy one of the leading positions in the production of "big data". The ability to spread and share messages, photos, music, videos with friends, and create and conduct various events, including for the purpose of business promotion - all this represents a huge amount of constantly generated, aging, updated data. Large amounts of data, including social networks data, as well as the relationships (links) between them must be presented in the form convenient for perception [1-6].

Often, when it comes to objects representing a network, for example, a social one, the notion of data visualization is closely related to the notion of graphs. The network represented as a graph is simple for perception and further analysis. An important task is to represent links in social networks to identify various kinds of dependencies.

2. Collecting data from a social network

To represent a social network in the form of a graph, many different tools and tools can be used. In the framework of this work, the following was used to solve this problem: an application developed in C # that allows obtaining the necessary data and performing their analysis; a tool for visualizing Gephi data for graphically representing the graph of dependencies (the so-called graph of the user's friends).

The software itself is visually an authorization form on which the user's login and password of the user account are entered (figure 1).

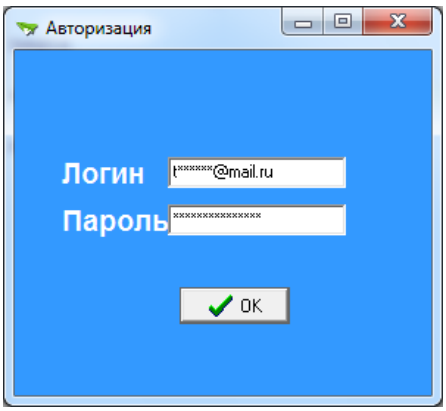

Fig. 1. Software interface.

After entering the login and password, the authorization of the user in the social network and access to the necessary information, namely: to the list of the user's friends, the list of communities, photos, messages, etc., takes place via the open OAuth authentication protocol version 2.0. In the framework of this work, we were interested in the possibility of extracting the user's friends from the social network, so the remaining items were ignored.

Each user of the social network has a unique identifier, or else an ID, which allows you to uniquely identify the user. Using the properties of the built-in API of the social network “Vkontakte”, you can, knowing the user ID, extract information about his friends, up to nesting level N. In other words, you can extract a list of friends (N = 1), friends of friends (N = 2), etc. We were interested in the list of friends up to the level of nesting N = 2.

The list of friends extracted from the social network and converted, takes the form of a text file containing the user ID, authorized in the social network and then in the tabular form of the user ID and his name (full name). Knowing the user's ID, you can also find out the list and his friends, which is similarly recorded in the file. An example of the output file part by user and each of the user-friends is shown in figure 2.

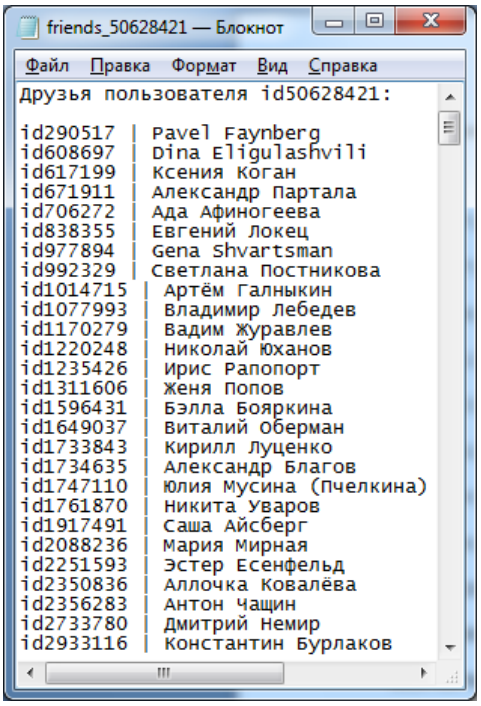

Fig. 2. An example of the file, which contains the information about the friends of the user.

Further, by concatenating the files, a common list of all friends is obtained, along which a list of all friends (dimension K) is constructed, from which an adjacency matrix (of dimension K × K) is constructed, on which the dependency graph is subsequently built, by the Gephi software.

An adjacency matrix is a K × K dimension matrix containing a list of friends horizontally and vertically, and a row of 0 or 1 at the intersection of the row and column. The contents of the cell of the table matrix is 0 if the users are not familiar (not included in the list of common friends between the user and the friend of the user) and 1 otherwise if there is a "friendship" relationship between the specified users. After construction, this matrix is saved in the .csv format for further loading into Gephi. An example of an adjacency matrix is shown in figure 3.

	Air Sola	Ildar Khalitov	Igor Rytsa	Svetlana S	Alexey Sa	Anastasiya	Andrey M	Maksim R	Alexsande	Yuri Nagul
Air Sola	0	0	0	1	0	0	0	0	0	1
Ildar Khalitov	0	0	0	0	0	0	0	0	0	0
Igor Rytsarev	0	0	0	0	0	1	1	1	1	1
Svetlana Sukhanova	1	0	0	0	0	0	0	0	0	0
Alexey Satonin	0	0	0	0	0	0	0	1	0	0
Anastasiya Kireeva	0	0	1	0	0	0	1	1	1	1
Andrey Mukhataev	0	0	1	0	0	1	0	1	1	1
Maksim Raguzin	0	0	1	0	1	1	1	0	1	1
Alexsander Nagulov	0	0	1	0	0	1	1	1	0	1
Yuri Nagulov	1	0	1	0	0	1	1	1	1	0

Fig. 3. Matrix of adjacency of the graph of friends.

3. The construction of a graph, classification of vertices, finding the most significant vertices

Constructed on the basis of the list of all friends, the contiguity matrix of the future graph is loaded into the Gephi software tool, in order to further visualize the graph of dependencies. Gephi is a software product for network analysis and visualization of data, written in the high-level Java language [7]. The constructed Gehpi graph looks like this (figure 4):

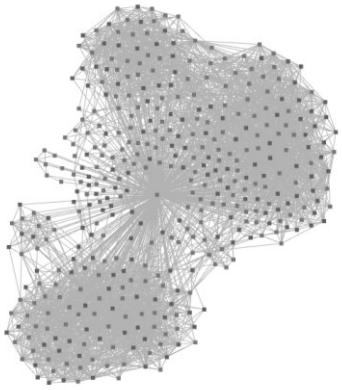

Fig. 4. The graph of users dependencies.

In this graph, the vertices are users of the social network, and the edges are the relation "friendship" between users.

It is worth noting that friends of friends of the user who do not have common relations with the user did not interest us in the framework of this work, so these tops-friends of friends were deleted from the graph.

The next step is to classify the vertices of the graph. The following classification is proposed in the paper:

- core is a vertex containing at least μ vertices in an ε-neighborhood
- hub is a separate node whose neighbors belong to two or more different clusters;
- outlier - this is a separate vertex, all neighbors of which belong to the same cluster, or do not belong to any cluster [8].

To implement such a classification, the SCAN algorithm is used [9].

The principle of the SCAN algorithm is described below.

Search begins with an initial visit of each vertex once [8], in order to find structurally-connected clusters, and then visit isolated vertices to identify them (hub or outlier).

SCAN performs one network pass and finds all structurally-related clusters for a given parameter. In the beginning all vertices are marked as unclassified. The SCAN algorithm classifies each vertex as either a member of a cluster, or as not being a member [5]. For each vertex that is not yet classified, SCAN checks whether this vertex is a kernel. If the vertex is the core, the new cluster expands from this vertex. Otherwise, the vertex is marked as not being a member of the cluster.

To find a new cluster, SCAN starts with an arbitrary kernel V and looks for all vertices that are structurally reachable from V. This is quite enough to find a complete cluster containing the vertex V. A new cluster ID is generated, which will be assigned to all found vertices.

SCAN begins by setting all the vertices in the ε-neighborhood of the vertex V in the queue. For each vertex in the queue, all directly reachable vertices are calculated, and those vertices that have not yet been classified are inserted into the queue. This is repeated until the queue is empty [8].

Vertexes that are not members of clusters can be additionally classified as hubs or extraneous. If a single vertex has edges on two or more clusters, it can be classified as a hub. Otherwise, it's an outsider.

A distinctive feature is the presence of parameters and that can be set by the user or expert. In this case, the optimal value of these parameters can be found by machine learning the system using certain network segments.

Since Gephi is an opensource platform [7], one of its great advantages is the ability to write its own modules that implement various algorithms. Thus, using the algorithm from [9], a module that implements the SCAN algorithm was written.

4. Results and Discussion

The result of the SCAN algorithm work on the graph, constructed in Gephi, is a text file containing a list of the user's friends and typing it as the top of the graph (figure 5).

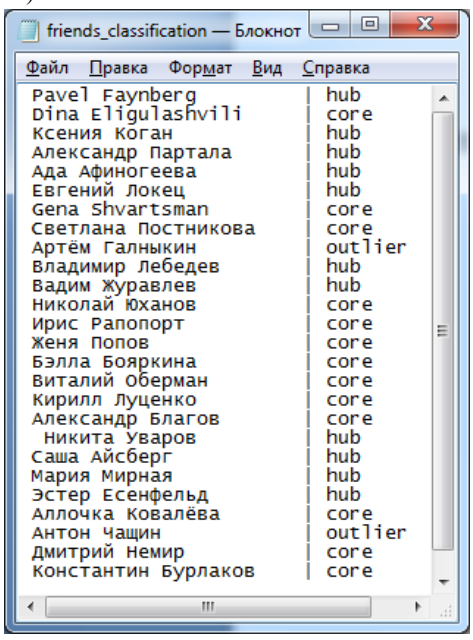

Fig. 5. The results of SCAN-algorithm.

It is worth noting that the SCAN algorithm has a certain limitation on the dimension of the graph used. On graphs with high dimensionality (N> 500) there is an error in the work.

One way to solve this problem is to modify the algorithm for parallel computations [10]. The idea of this modification is to split the sets of communities into subsets between processors. However, one should take into account that for balancing these subsets should have roughly the same sum of squares of community sizes.

The authors set the task of creating a distributed modification of the SCAN algorithm for ultrahigh-dimensional graphs.

5. Conclusion

Social networks and connections in them are the subject of research in this research work. The approach based on the representation of social networks in the form of graphs, makes it possible to apply algorithms for clustering graphs of high dimension. The algorithms described in the paper make it possible to classify segments of a social network, and also to find elements of the greatest interest, for example, users that affect all elements of the same community. These algorithms are planned to be finalized for subsequent application in solving practical problems of finding communities in the segment of social networks in the Samara region.

The Gephi tool, which makes it possible to implement visualization of social networks, was explored, and a software tool that allows to present data in the form required for research was developed.

Acknowledgements

The work has been performed with partial financial support from the Ministry of Education and Sciences of the Russian Federation within the framework of implementation of the Program for Improving the Samara university Competitiveness among the World's Leading Research and Educational Centers for the Period of 2013-2020s.

References

[1] Tan W, Blake MW, Saleh I, Dustdar S. Social-network-sourced big data analytics. IEEE Internet Computing 2013; 5: 62–69.
[2] Blagov A, Rytcarev I, Strelkov K, Khotilin M. Big Data Instruments for Social Media Analysis. Proceedings of the 5th International Workshop on Computer Science and Engineering 2015; 179–184.
[3] Ivanov PD, Lopukhovsky AG. BigData technologies and various methods of representing big data. Engineering Journal: Science and Innovation, 2014; 9.
[4] Agafonov AA, Myasnikov VV. Method for the reliable shortest path search in time- dependent stochastic networks and its application to GIS-based traffic control. Computer Optics 2016; 40(2): 275–283. DOI: 10.18287/2412-6179-2016-40-2-275-283.
[5] Popov SB. The Big Data methodology in computer vision systems. CEUR Workshop Proceedings 2015; 1490: 420–425. DOI: 10.18287/1613-0073-2015-1490-420-425.
[6] Ilyasova NYu, Kupriyanov AV, Paringer RA. Formation of features for improving the quality of medical diagnosis based on discriminant analysis methods. Computer Optics 2014; 38(4): 851–855.
[7] An open platform for presenting data in the form of graphs GEPHI. URL: https://gephi.org (13.02.2017).
[8] Khotilin MI, Blagov AV. Visual presentation and cluster analysis of social networks. Information Technologies and Nanotechnologies, 2016; 1067–1072.
[9] Xu X, et al. Scan: a structural clustering algorithm for networks. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007; 824–833.
[10] Drobyshevskiy MD, Korshunov AV, Turdakov DY. Parallel modularity computation for directed weighted graphs with overlapping communities. Proceedings of the Institute of System Programming RAS 2016; 28(6): 153–170.