French-Azerbaijani University

Network and Algorithms

report on project

Analysis of the user connections in a Social Network

Group:

CS-016

Authors:
Emin Sultanov
Sanan Najafov
Imran Ibrahimli

January 3, 2020

Contents

1 Introduction

This report looks into the analysis of the connections between users in an example of **VKontakte** social network. The report also describes the procedure of working the program that gathers information from the users who have the connections between each other using **`VK API`** and represents it as one data. Purpose of this project, data analysis, working process, the result will be also discussed in this report.

2 Project Description

The main and the most important idea of the project is to understand how every social network works. The second idea is the analysis of the graph by calculating and finding important users according to different centralities. After finding them it is possible to create the shortest path between the user on the board of the network, which means that the degree connectivity of this user is equal to one, and one of the important users. In this case, the shortest path is needed to know from which user and which way to move to the main user, as the connection with him may give the possibility to connect with other users in the network.
Another purpose is to analyze the information about each user in the network obtained after the compilation of the program to show an average user. The information about the average user is needed to show what a person should be in order to be accepted into this network by others. Moreover, from this analysis, it is possible to deduce the approximate information about each user in the network and especially about the initial user from whom the research begins.

3 Data Description

Operational data obtained through the analysis of friendly relations between the users after the program is compiled. The resulting data consists of ten attributes:

- **User ID** - `string` value. The id of a target user which used for analysis.
- **Friend ID** - `string` value. The id of the target user's friend obtained after analysis.
- **Friend's first name** - `string` value. The first name of the target user's friend.
- **Friend's last name** - `string` value. The last name of the target user's friend.
- **Friend's sex** - `string` value. The value can be `'0'` (unspecified), `'1'` (female) and `'2'` (male).
- **Friend's birthday** - `string` value. Friend's full birth date written in *DD.MM.YYYY* format. The result can be obtained as `'0'` if the birth date is hidden from public or in *DD.MM* format in case if birth year is hidden from public.
- **Country** - `dictionary` value. The country id where the user's friend is currently located along with its name. The value is `'0'` if the country is not specified.

- **City** - `dictionary` value. The city id where the user's friend is currently located along with its name. The value is `'0'` if the city is not specified.
- **Universities** - `list of dictionaries` value, where each `dictionary` contains
  - **id** - university ID
  - **country** - ID of the country the university is located in
  - **city** - ID of the city the university is located in
  - **name** - university name
  - **faculty** - faculty ID
  - **faculty_name** - faculty name
  - **chair** - university chair ID
  - **chair_name** - chair name
  - **graduation** - graduation year
- **Friend's personal info** - `dictionary` value that contains:
  - **political** - political views
  - **langs** - languages
  - **religion** - world view
  - **inspired_by** - inspired by
  - **people_main** - important in others
  - **life_main** - personal priority
  - **smoking** - views on smoking
  - **alcohol** - views on alcohol

***Note** *:* all the values of the keys in the dictionaries are the `string` values

4 Creation Process

4.1 Algorithm

The first step of the creation of this project is the development of the algorithm for the program. The algorithm for this program consists of two parts.
The first part involves the obtaining of the working data. In order to get the information from the users, it is necessary to know the initial user from whom it is supposed to start the investigation. As there is a huge number of users in each social network, it is supposed to add three restrictions such as the maximum depth of propagation, the maximum number of connections in order to reduce the time of information collection and the maximum number of connections per user to increase the variety of users. It is important to check each of the restrictions during each investigation to not exceed the limit. Knowing that several users can have

direct connections with each other, it is also necessary to remember which connection was already visited in order to not infinitely repeat the process going from one user to another and in reverse direction. After each research process, the information should be kept in one place. When the limit of the maximum number of connections is reached, it is important to go through all the users on the board (with degree connectivity equal to 1) and check if they have connections with others in the created network.
The second part involves the graph and data analysis. As one of the goals of this project is to get the shortest path from the board user to the important users it is supposed to find firstly the important users. Knowing that the centrality values determine the importance of the node in the graph it becomes necessary to find them. After finding the shortest path it is needed to analyze the data in order to achieve the second purpose of the project. The data analysis involves finding the person with the average personal qualities according to the network. To accomplish this goal, it is assumed that we need to view all the people in our data, select only the different users, and select all the attributes with the highest usage among them.

4.2 Program

Before executing the program it's important to install all the dependencies. The following libraries were used in the project:

- `vk` - to work with vk api.
- `unicode` - to rewrite the words in English (e.g. François -> Fransua).
- `yandex.translate` - to translate the words from a different language in English.
- `pandas` - to work with data.
- `numpy` - to use numpy functions (arrange and etc.) and variables (infinity).
- `matplotlib` - to represent graphs and bar charts.
- `networkx` - to work with graphs.

4.2.1 Preliminary Preparation

All the dependencies were specified in the `requirements.txt` file that can be used to install the necessary libraries:

- `sudo pip3 install -r requirements.txt`

In order to accomplish the project, it is supposed to work with VK API. Therefore it is necessary to get access to their API. Steps to get API access:

1. register an account in VKontake (`link to VK`)
2. create standalone application (`link to create an application`)

3. Go to the settings (located on the left side of the page)
4. Remember the App ID (located on the top of the settings section)
5. change the Open API in the settings to enabled and write localhost as website and base domain, save the changings
6. Open new window and write in address bar:
   https://oauth.vk.com/authorize?client_id=**YOUR APP ID**
   &display=page&redirect_uri=https://oauth.vk.com/blank.html
   &scope=friends,offline&response_type=token&v=5.103
7. Allow access to your account for this application
8. Get the access token from the address bar:
   https://oauth.vk.com/blank.html#access_token=**YOUR ACCESS TOKEN**&expires_in=0&user_id=**some user id**

Note *: **strongly suggested not to try to compile the program many times at one time or after a short pause to not exceed the API usage limit and not be banned!**

4.2.2 Coding

The second step of the working process is the creation of the program. According to the algorithm, for the first part, it is supposed to gather data. For this purpose, it is necessary firstly to add obtained token into the `settings.py` (e.g. `token = ('your token')`). Afterwards, it is possible to start the VK API session and gathers the data. Data collection achieves by using three functions:

- **`make_list()`** - the function takes as input the `id` of the user to investigate for the friendship connections, list of found `connections` with the found friend's attributes (id, name, surname, etc.), list of already `visited` connections between the users, maximum number of connections, maximum depth, maximum number of connections per user and the current depth. This function works on the principle of the deep first search algorithm as it is using recursion. After each calling this function, the program checks the limit of the connections. If the number of the connections is not exceeded, it investigates the friend list of the user with given id by calling the function `api.friends.get()`, adds all the possible connections between the user and user's friend with his attributes to the list of connections by using the function `add_to_list()`, goes deeper if the depth is not exceeded and tries to investigate the user's friend for his friendship connections while checking the found connections in the list of already visited. In case if the page of the user with the given id is private, blocked, deleted or simply can not be accessed it skip the investigation of this user.
- **`add_to_list()`** - the function takes as input the `id` of the user that is investigated, list of found `connections` with the found friend's attributes (id, name, surname, etc.), list of the found `friends` after the investigation, current depth, the maximum number of connections, the maximum number of connections per user. After each calling this function, the program goes through the list of founded

friends and their attributes, checks if the friend is not deleted permanently and adds all possible connections into the list of connections taking account of the restrictions. If the investigated user is the initial user, which means the depth is equal to 0, it adds the maximum found connections.

- **`connect_border()`** - the function takes as input the obtained `connections` between users and users' friends with their attributes and the list of already `visited` connections. After each calling this function, the program searches for not investigated users in the list of connections and tries to find any connection with other users while checking if the connection is already visited.

When the data is collected, the implementation of the second part of the algorithm begins. For this part, it is considered first to find the important users according to centralities values. For this purpose, it is needed to use the functions of the networkx library:

- **`betweenness_centrality()`** - returns the dictionary of betweenness centralities for each node. Betweenness centrality determined as the number of the shortest paths passing by the given nodes.
- **`closeness_centrality()`** - returns the dictionary of closeness centralities for each node. Closeness centrality is determined by closeness to all other nodes.
- **`degree_centrality()`** - returns the dictionary of degree centralities for each node. Degree centrality is defined as the number of links incident upon a node.
- **`eigenvector_centrality()`** - returns the dictionary of eigenvector centralities for each node. Eigenvector centrality is determined as a measure of the influence a node has on a network. If a node is pointed to by many nodes (which also have high Eigenvector centrality) then that node will have high eigenvector centrality.

After finding all centralities it is possible to find the shortest path between a random user on the board of the network and with all the nodes with the highest value of one of the centralities.
Data analysis is performed using one main function and three minor functions:

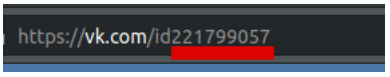
- **`get_person()`** - the main function that reads the data frame "user_connections.csv" that was created after data collection and starts the analysis of the data. Each analysis is performed by calculating the number of usages of the distinct instances of the different attributes. To do this calculation, it is supposed to create and use dictionaries for each attribute, where the value of the instance increments after obtaining two or more identical instances or equates to one if the instance appears for the first time in the dictionary. In case if the analysis of the attribute gives out several instances with the same highest value it is considered to choose randomly one of them. To avoid different symbols, emojis in the achieved text during the analysis of "inspired_by" attribute it is suggested to use `remove_emoji()` and `re.sub()` functions. It is also considered to translate the achieved expressions from the user's personal information to get more precise analysis. The translation is achieved by using `YandexTranslate(yandex_key).translate()`.
- **`remove_emoji()`** - the function takes as input the phrase with emoji symbol and removes it.

- **`max_items()`** - the function takes as input dictionary with instances as keys and the number of their appearance in user profile as values and returns the list of keys with the same highest number of the usage.
- **`max_four_languages()`** - the function takes as input the dictionary of the different languages as keys with the number of their appearance in the user profile as values and returns four the most used languages.

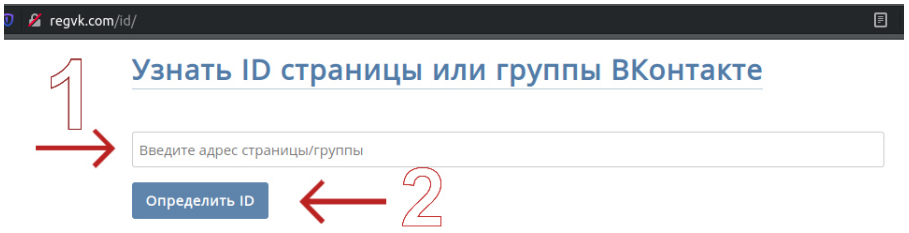
5 Implementation

In order to start the compilation of the program, it is necessary to give four values:

- Initial **id** of the user to start the investigation. The id number of each user is almost always written at the end of the link to profile:



  Or it is also possible from the web application by passing through `the link` and writing firstly the link to the page in the empty string and secondly by clicking the button:



- **The maximum number of connections**, where the possible inputs are -1 (to get all possible connections), 0 (to use a default value, 1500) and any other positive number.
- **The maximum number of connections per user**, where the possible inputs are -1 (to get all possible connections from one user), 0 (to use a default value, 50) and any other positive number.
- **The maximum depth**, where the possible inputs are -1 (without restriction by the depth), 0 (to use a default value, 2) and any other positive number.

6 Results and Analysis

In order to represent the graph, it is supposed to use functions of the networkx library by indicating user id on each node and matplotlib library.
As an example, it was considered to take id equal to 53083705 (ID of Dmitriy Medvedev) and default values for other inputs. The obtained graph is below:

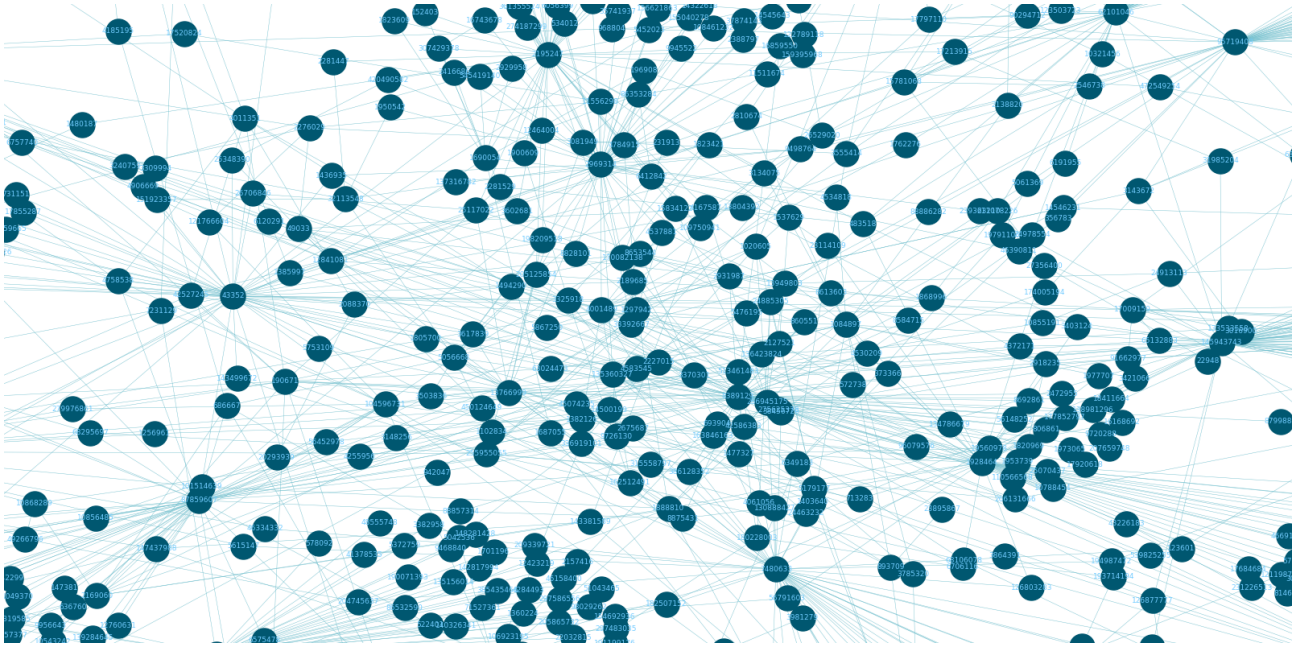


Figure 1: A fragment of the graph representation.

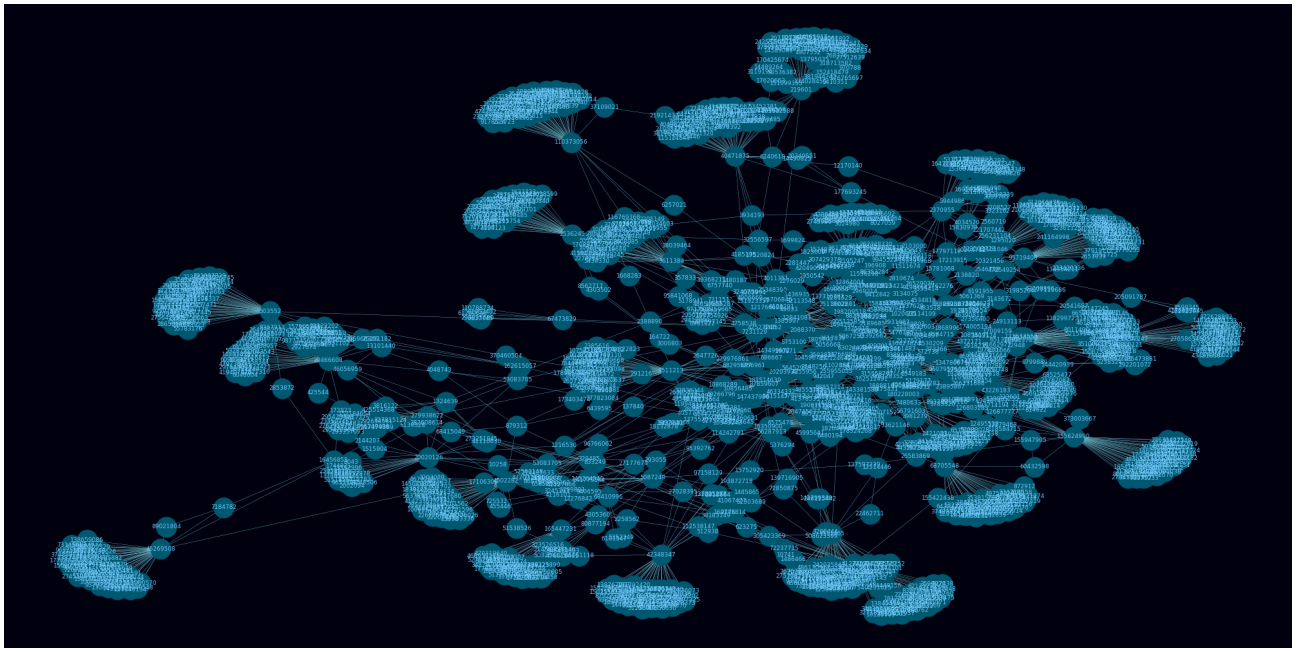


Figure 2: Full graph representation.

The graph fragments representation according to the different centralities:

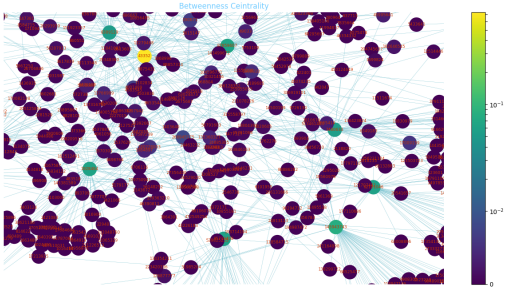


Figure 3: Betweenness Centrality representation.

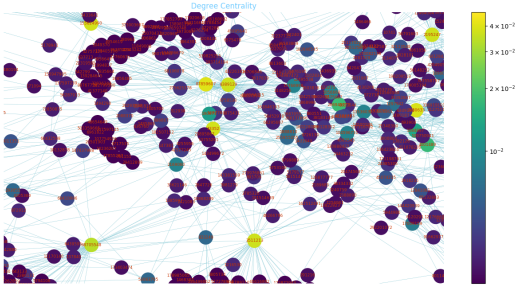


Figure 4: Degree Centrality representation.

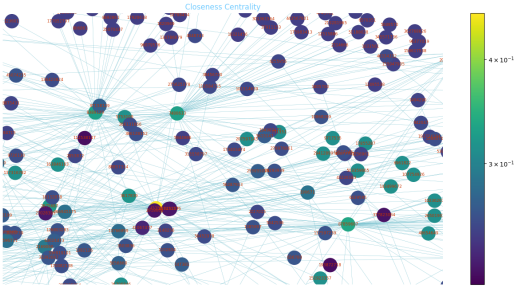


Figure 5: Closeness Centrality representation.

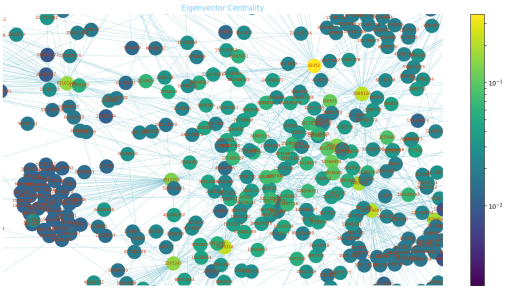


Figure 6: Eigenvector Centrality representation.

After finding centrality values for each node, it is possible to deduce, that the user with ID 43352 is an important user according to betweenness, closeness and eigenvector centralities and the user with ID 145943743 is an important user according to degree centrality. Afterward, it became easy to find the shortest path from a randomly selected board user (in our example the user with ID 223564190 was selected) to the important users.
The resulting path from the 223564190 to 43352 is: 223564190->503552->43352
The resulting paths from the 223564190 to 145943743 is: 223564190->503552->43352->145943743
The shortest paths from the random user to the important users:

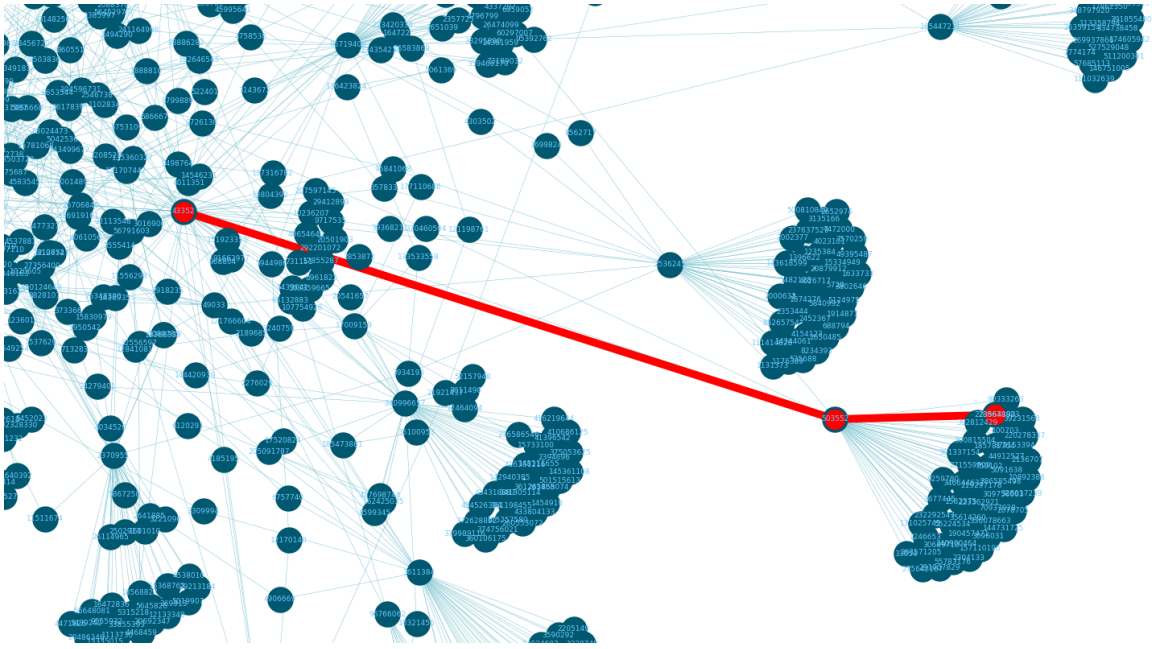


Figure 7: The shortest path from 223564190 to 43352.

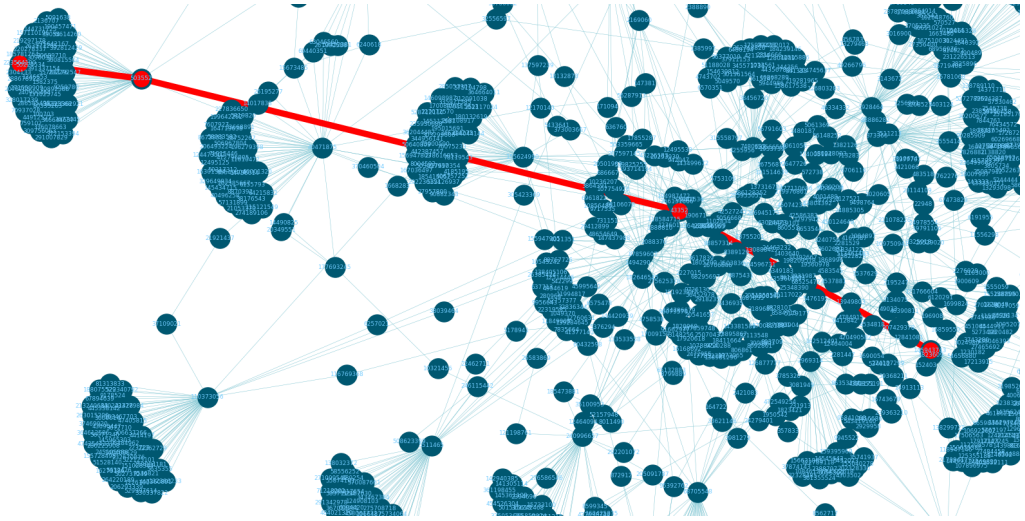
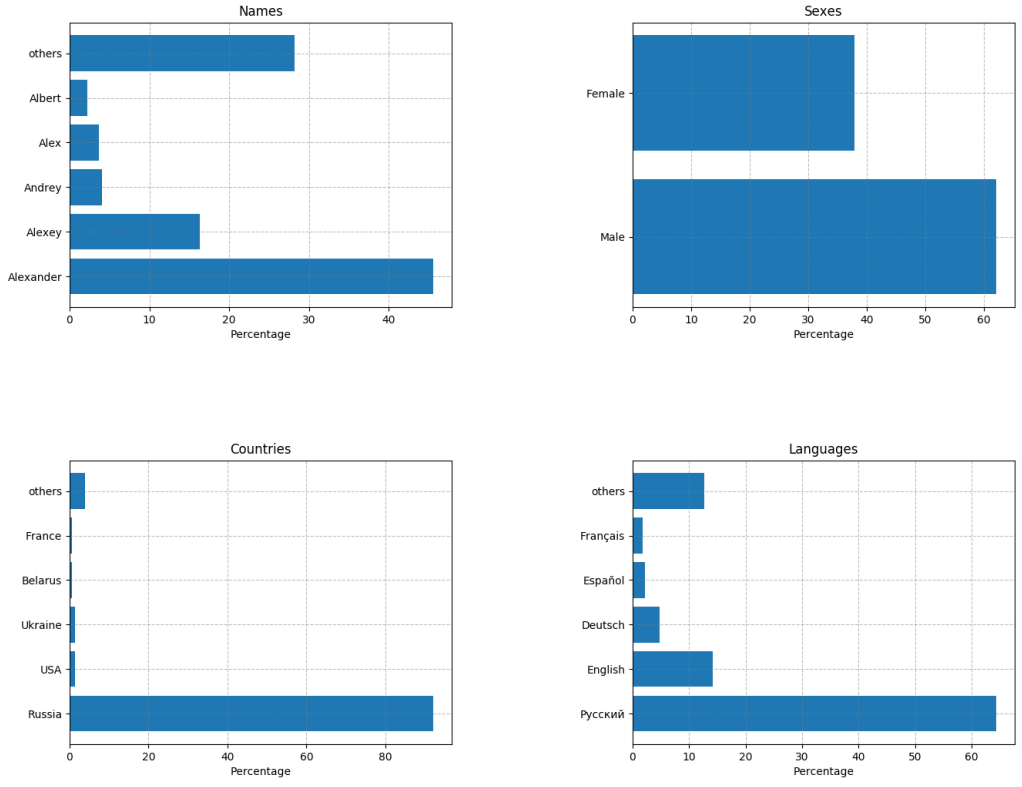


Figure 8: The shortest path from 223564190 to 145943743.

The analysis of the data is represented in the bar charts, some of them:



When all the instances with the highest values of appearance are found, the result is printing on the screen:

Name with the highest usage : Alexander
Surname with the highest usage : Ivanov
Average sex : Male
Average birthday : 1.4.1987
Average country : Russia
Average City : Moscow
First four languages with the highest usage : ['Русский', 'English', 'Deutsch', 'Español']
Most common political views : Moderate
Most common religion : Orthodoxy
Most common inspiration : Music
Main in people : Kindness and honesty
Main in life : Family and children
Average relation to smoking : Negative
Average relation to alcohol : Negative
Most common university name : МГК им. Чайковского
Most common faculty name : Orchestra Faculty

From obtained results, it is possible to deduce the most common attributes and qualities of the person in the network and the path to get the connection with the main users in the network as it was intended to find at the beginning of the project.

7 Conclusion and Perspectives

The average personality is the necessary information that is needed to make as many connections as possible in the investigated network since the information of the user's profile creates the first impression about the person itself and can give an evaluation of how much the person is right or show what should a person be like to become right for this network or even society. Moreover, the achieved average personality can also give an impression of the investigated person in case if it is necessary to know his preferences according to his friends in advance before sending a request for friendship. Another perspective of this project is to find the right direction of the step-by-step creation of links with users until the link with the main user is created. Direct link with the main user allows better and faster communication with others taking into account that the main user has more connections and may have more impact on the network.
In conclusion, according to the perspectives, it is possible to say that the algorithm of this project can be used in all social networks and dating sites if there is access to people that are liked or followed by the users.

8 References

List of web-sites that contains useful information to accomplish this project:

- https://pypi.org/project/yandex-translater/1.0/
- https://gist.github.com/slowkow/7a7f61f495e3dbb7e3d767f97bd7304b
- https://networkx.github.io/documentation/stable/index.html
- https://habr.com/ru/post/319178/
- http://www.dimensions-math.org/Dim_regarder_E.htm
- https://vk.com/dev/methods
- https://www.geeksforgeeks.org/
- https://stackoverflow.com/
- https://translate.yandex.com/developers/keys