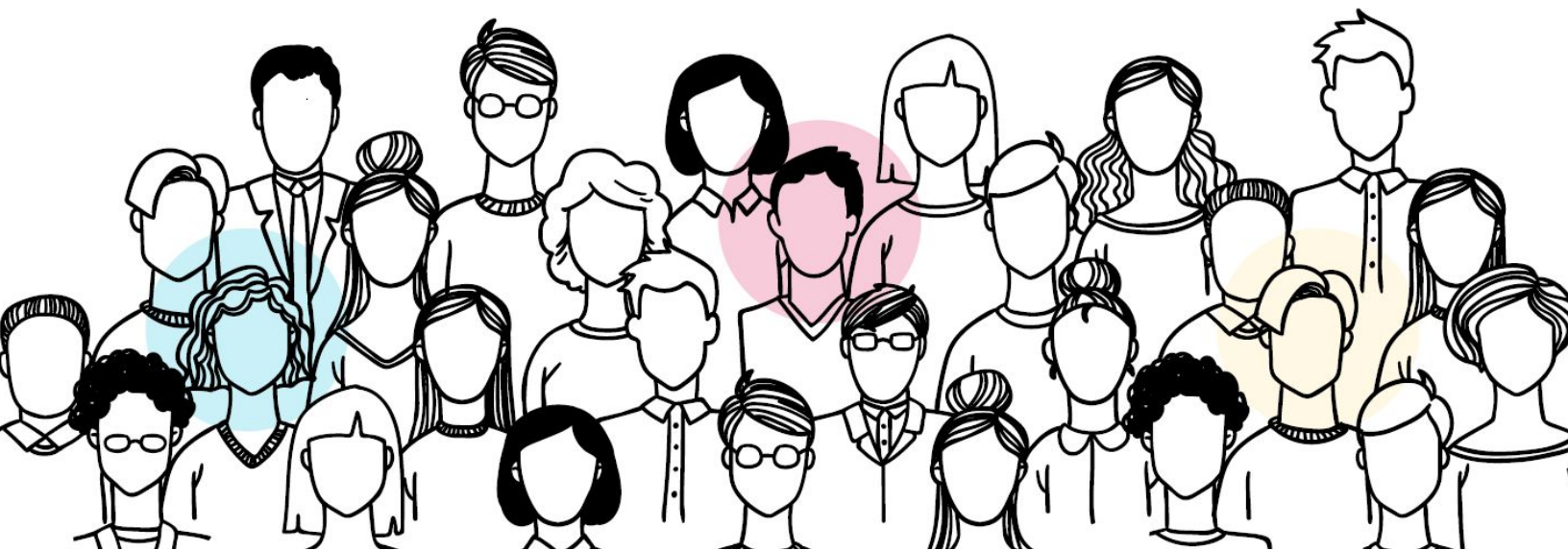


Graph Analysis and Social Networks

# VK FRIEND NETWORK ANALYSIS

## Community Detection

Sofia Smolianinova  
Master's Programme in Data Science  
Erasmus



# Abstract

Community detection is an important area in Social Networks Analysis. Research has shown that discovering the structure of the social network and detecting communities can be used in sociology, biology and computer science because systems are often represented as graphs. Nowadays, the most popular social networks like Facebook have a huge size and reach billions of nodes and the complexity of this task is growing.

This study aims to determine which groups are present in my VKontakte social network graph. Building on existing work on analysis of the VKontakte social network, the article answers: How to get relevant data from VK API and visualize this data? In this context, data means the tree of the friends and their respective friends for a specific user.

The main goal of this paper is to give a comprehensive description of the community detection algorithm for VK social graphs. For this I provide a python language code to extract the data from VK API and convert it to a .gexf file format. In addition, I used the Gephi tool to visualize the communities and relations between people. Finally, I provide further research directions as well as some open challenges.

# Introduction

In the scope of this assignment I want to present an analysis of my VK friend network. The first question that comes to the mind: **What is VK?**

VK or originally VKontakte is the second largest social network in Europe after Facebook. The name of this social network means “keep in touch”. It is available in several languages, but is especially popular among Russian-speaking users around the world, especially in Russia, Ukraine, Kazakhstan and Belarus. As I’m living in Russia the most representative social network to complete the community detection task for me is VK. As Facebook or other social networks, VKontakte allows users to create pages, groups, public pages (for artists or famous peoples) and events. Also you can share and tag images, audio and video files and play browser-based games. VKontakte is pretty similar to Facebook in its functionality and nature of usage. The platform offers few additional features to its audience such as custom audio playlists creation; ability to share videos of unlimited length and super detailed inner search capabilities.

I have been using this social network for more than 9 years. Therefore, it's a great choice for researching the community's detection in a social graph.

# 1. Methods

In this chapter we provide an algorithm for data extraction from VK API and conversion to a .gexf file format. In addition, the second part contains steps to visualize the communities and relations between people via Gephi tool.

## 1.1. How to get data?

In this particular section we define a detailed algorithm for obtaining data about friends of a particular user. Also, we can define the depths for the friends tree, hence, be careful: the number of requests to VK API is limited per day. And for example if you have a huge number of friends as well as your friends you can receive an error.

### 1.1.1. Creation and authorization of the application

To get access to the VKontakte API, we need to create a Standalone-application, after which we can use built-in API methods. To create a Standalone-application we can use the following [link](#) and just specify the name of the application and click the blue button “Add application”.

The screenshot shows the VK Developers interface for creating a new application. The main form has a title field containing "Graph Mining and Social Network Analysis". Under the "Платформа:" (Platform) section, the "Standalone-приложение" (Standalone application) option is selected. A blue button labeled "Подключить приложение" (Connect application) is located below the platform selection. The left sidebar lists various developer resources. The top navigation bar includes links for "Продукты", "Документация", "Мои приложения", and "Поддержка", along with a search bar. The footer contains copyright information and links for "О ВКонтакте", "Помощь", "Правила", "Реклама", "Разработчикам", and "Вакансии".

When the application is created on the Settings Tab you can find the application ID that we will use to get an access\_token. Next, we need to authorize our application. This process consists of 3 stages:

### Step 1: User Authentication on VK

To start authorization you need to create a browser window and open an authorization dialog in it with response\_type = token parameter at:

```
https://oauth.vk.com/authorize?  
client_id=APP_ID&  
scope=PERMISSIONS&  
redirect_uri=REDIRECT_URI&  
display=DISPLAY&  
v=API_VERSION&  
response_type=token
```

- APP\_ID — your application ID
- PERMISSIONS — requested application access [permissions](#)
- DISPLAY — authorization window appearance, the following options are supported: **page**, **popup**, **touch** and **wap**.
- REDIRECT\_URI — URL where access\_token will be passed to
- API\_VERSION — [API version](#). Before increasing the version, please, read a [changelist](#).

### Step 2: Providing Access Permissions

After successful login on the site, the user will be prompted to authorize the application by allowing access to necessary settings, requested using scope parameter.

### Step 3: Receiving "access\_token"

After successful application authorization, user's browser will be redirected to REDIRECT\_URI, URL specified when the authorization dialog box was opened. With that, access\_token access key for API and other parameters will be passed in URL part of the link:

`http://REDIRECT_URI#access_token= 53..6a3&expires_in=86400&user_id=8492`

Together with the access\_token key its expires\_in life time in seconds will be specified. In case of authorization error, information about this error will be passed as GET parameters to REDIRECT\_URI. After successful authorization you can make API requests.

### 1.1.2. Python application to represent graph in .gexf file format

The aim of the Python application is to extract the data from VK API and create a file with a social network graph structure in an appropriate format. Check [github repository](#) to download source code and all required data.

This application will help to get data from VK API and save all necessary information as a tree of friends for further work. To obtain the information python application used two methods: friends.get and users.get.

Below you can see an example of the tree of friends file in JSON format.

```
[
  {
    "id": 76035331,
    "name": "\u0410\u043b\u0435\u043a\u0441\u0430\u0441\u0430\u0434\u0434\u0440\u0430\u0441\u0441\u0442\u0430\u0435\u043b\u0444\u0444\u0434\u0434\u0435\u0432\u0432\u0430",
    "sex": 1,
    "friends": [
      {
        "id": 1982443,
        "name": "\u0418\u0438\u0445\u0438\u0430\u043b\u0441\u0441\u0440\u0435\u0447\u0430\u0430",
        "sex": 2
      },
      {
        "id": 4661851,
        "name": "\u0422\u0435\u0444\u0434\u0430\u0430\u0441\u0442\u0442\u0440\u0443\u0430\u0435\u0432\u0430",
        "sex": 1
      }
    ]
  }
]
```

After all data manipulations application saves the graph structure to a file with the .gexf extension.

Requirements:

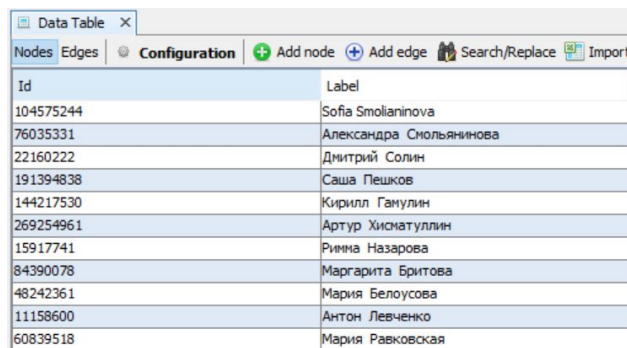
- Json
- Random
- Time
- Os
- Networkx
- vk\_api

## 1.2. Gephi tool for community detection task

First, we need to load the file in .gexf format from the previous step. The file with my social graph contains 178 nodes (177 friends + one root user) and 1642 edges that represent the relationships between people.

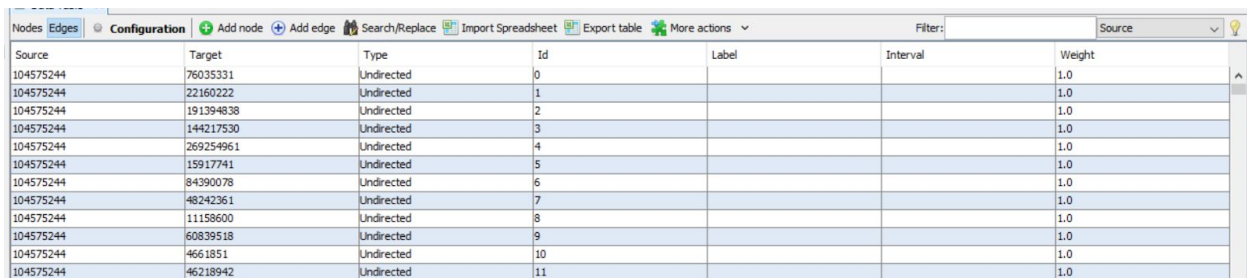
### 1.2.1. Gephi Data representation

On the Data Laboratory panel we can see data for Nodes and Edges. The Data Laboratory gives you the raw data as a table. Each node is described by ID - unique identifier for the node (ID in VK) and Label - name of the friend.



Id	Label
104575244	Sofia Smolianinova
76035331	Александра Смольянинова
22160222	Дмитрий Солин
191394838	Саша Пешков
144217530	Кирилл Ганулин
269254961	Артур Хисматуллин
15917741	Римма Назарова
84390078	Маргарита Бритова
48242361	Мария Белоусова
11158600	Антон Левченко
60839518	Мария Равковская

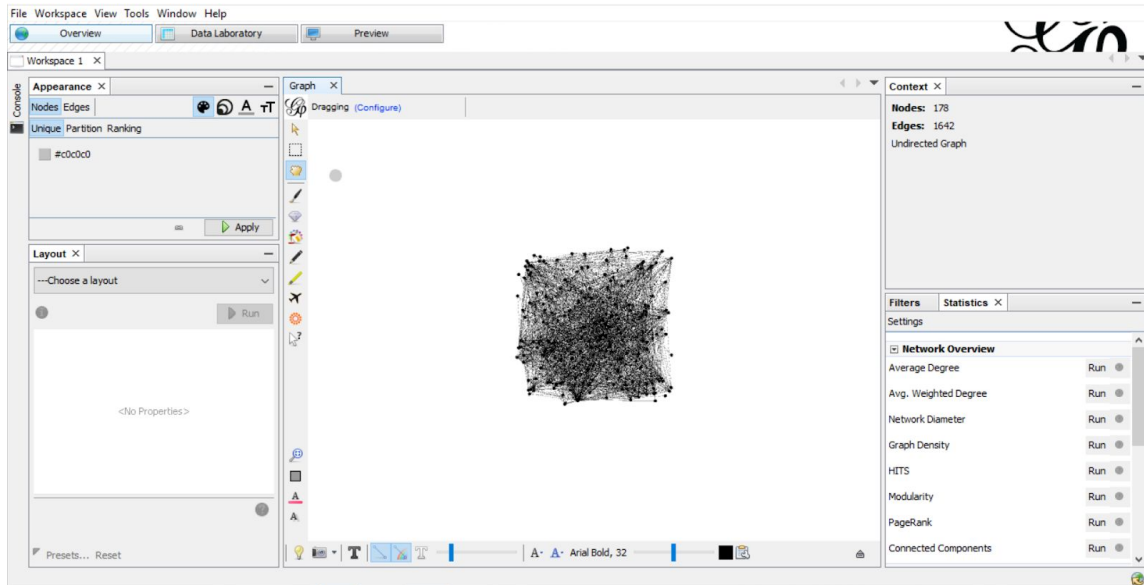
On the Edges tab the main columns are : Source (relation From), Target (relation To), Type - Undirected.



Source	Target	Type	Id	Label	Interval	Weight
104575244	76035331	Undirected	0			1.0
104575244	22160222	Undirected	1			1.0
104575244	191394838	Undirected	2			1.0
104575244	144217530	Undirected	3			1.0
104575244	269254961	Undirected	4			1.0
104575244	15917741	Undirected	5			1.0
104575244	84390078	Undirected	6			1.0
104575244	48242361	Undirected	7			1.0
104575244	11158600	Undirected	8			1.0
104575244	60839518	Undirected	9			1.0
104575244	4661851	Undirected	10			1.0
104575244	46218942	Undirected	11			1.0

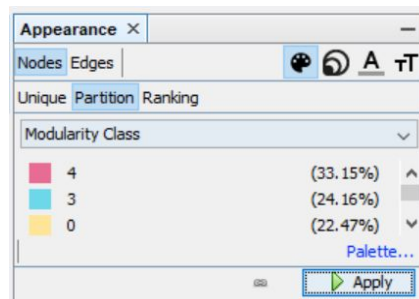
On the Overview panel we can see the first representation of the social graph. At first glance, it might seem like a bunch of ants or a cobweb. In order to get visualized information and highlight/detect communities on the graph, we need to calculate several statistics and run several algorithms to obtain the best result.

**Step 0:** Visualization of the initial data

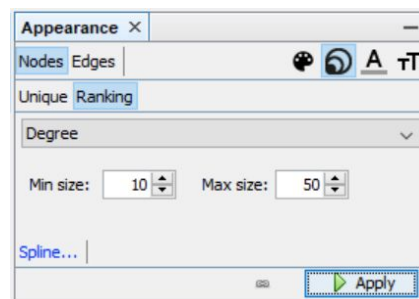


**Step 1:** Run Modularity statistics

**Step 2:** Change the color of the nodes based on the modularity statistics value. Nodes -> Color attribute -> Modularity class. By this way we detect communities and associate each community with a different color.



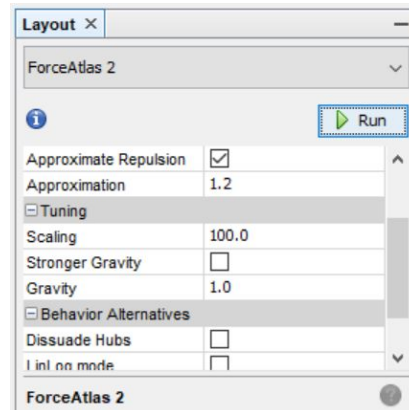
**Step 3:** Change the size of the node based on the degree of each node. Nodes -> Ranking -> Degree.



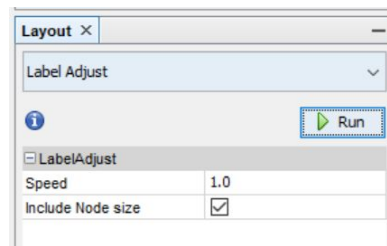
**Step 4:** Run Force Atlas 2 algorithm with parameters below to



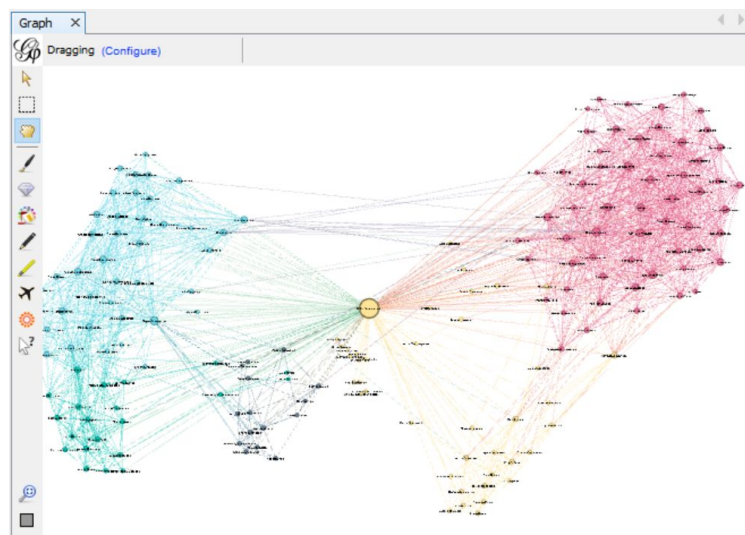
disperse nodes and get communities visualization.



**Step 5:** Run Label Adjust algorithm so that the labels of the nodes do not overlap.

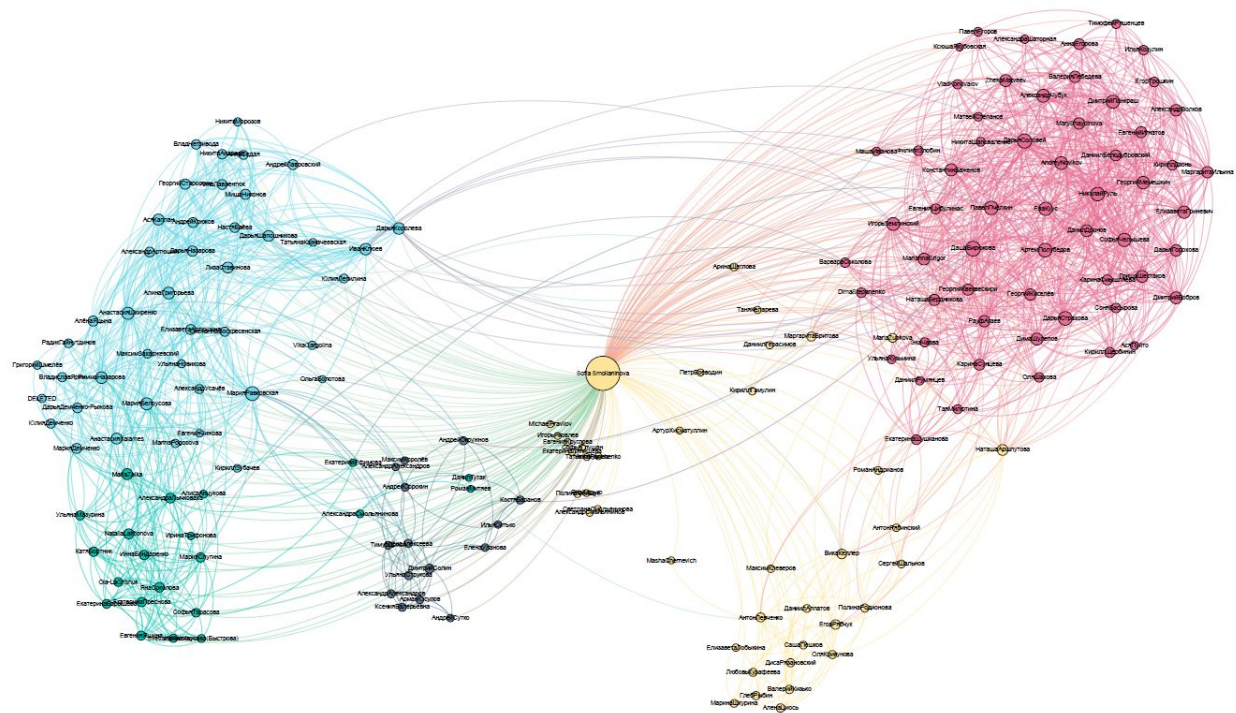


**Step 6:** We get the social network of my VK friends represented by the graph. The next step is to analyze identified communities. To analyze this step, we need expert help.



# Results

Once you have applied your community detection algorithm on your graph and you are satisfied with the result, it can be very interesting to analyze visualization. It helps to understand your community detection quality and get a global point of view on your communities and their members.



In this example, 5 communities were identified that we will consider in more detail. If we move from the lower left corner clockwise we will meet the following groups: Friends from dance classes (green), classmates and friends from school (blue), friends from the university clubs (pink), university classmates (yellow), friends from work in a football club (gray).

Analysis of the quality of the resulting communities can only be carried out with the help of an examination (of me), since for this it is necessary to know the data and real relationships between people. Since I come from a small town - Sosnovy Bor which is located near St. Petersburg, two communities: “dance club” and

“school friend” intersect. The city is small enough, many people know each other and it is difficult to draw a line and highlight some specific communities.

The next large group is “friends from the university” in a pink color. I study in the St. Petersburg Polytechnic University and often participate in organizing events and conferences. Therefore, I have many university friends who are somehow connected. This group is isolated from the previous two, since the city where people live is different. But you can also see that there are several connections from school friends to university ones, because some of my school friends entered the same university as me. Small yellow group consists of people from university: classmates and some random university friends. And the last group is friends from work in a football club.

## Discussion

In this work, only connections between people were used to identify communities. In the future, we can add the parameters of the city, age, place of work, likes and so on to detect the communities with better quality. All this data can be obtained using VK API. Also, we can expand the depth of the graph and use not only friends of friends data.

## Conclusion

To conclude this post, community detection in complex networks is still a very challenging field. Many algorithms have been created to efficiently identify communities inside large networks. However, correctly evaluating the performance of these algorithms is still an issue, especially in the case of overlapping communities.

We still saw with the example of the VK social network friend graph that the graph is small enough to build a nice visualization of our result and get an idea of the quality of the partition.

## References

- [1] "API methods | Developers | VK", Vk.com, 2020. [Online]. Available: <https://vk.com/dev/methods>. [Accessed: 29- Mar- 2020].
- [2] "GEPHI – Introduction to Network Analysis and Visualization", Martin Grandjean, 2020. [Online]. Available: <http://www.martingrandjean.ch/gephi-introduction/>. [Accessed: 29- Mar- 2020].
- [3] "Guide: Analyzing Twitter Networks with Gephi 0.9.1", Medium, 2020. [Online]. Available: <https://medium.com/@Luca/guide-analyzing-twitter-networks-with-gephi-0-9-1-2e0220d9097d>. [Accessed: 29- Mar- 2020].