Social media analysis of R and Python followers on Twitter

COSC2671 ASSIGNMENT 2

GROUP: A SMALL WORLD

DATE: 20 MAY 2018

Team members

- Chi Ting Low (s3611774@student.rmit.edu.au)
- Kanru Wang (s3559697@student.rmit.edu.au)
- Yong Kai Wong (s3615687@student.rmit.edu.au)

Dropbox link:

https://www.dropbox.com/sh/udhrz2gkd9zd1em/AACtF20rLU_ZFnp_vOO1nPita?dl=0

Google Drive link:

https://drive.google.com/drive/folders/16wgiwbBS5WRhtKH2u6RCimDGloqQt25v?usp=sharing

Table of Contents

1. Introduction

In recent years, there has been a debate among whether to use R or Python as open source tools for data science (Piatetsky, 2018). R is a language for statistical data manipulation and analysis that focus on delivering data analysis and graphical models on the user-friendly way (Matloff, 2013 and R Core Team, 2017). It is an open-source statistical environment developed by Rober Gentleman and Ross Ihaka in 1995 and is maintained by R core-development team (Gardener, 2012). Python was created by Guido Van Rossem in 1991 to increase programming productivity and readability (Theuwissen, 2018). Later, it was used by programmers for applied statistic and another statistic-related programming. Python is a flexible programming language; it can be used to program in procedural, object-oriented or functional (Summerfield, 2013). It is also a cross-platform language which is portable and can be run on all major platforms without program changes (Ascher & Lutz, 1999). While each tool has its advantages and disadvantages, both tools are relatively essential for the development of data science community.

The current project aimed to detect communities of Python or R users in the data science field on Twitter to answer the following hypotheses:

- Most Python users have the background of computer science
- Statisticians are more likely to use R for data analysis and graphical models compared to Python which is primarily used for web development and machine learning.
- Kaggle and DataCamp have a balanced proportion of Python and R users.

In this project, we used Python 3.6 (Van Rossum, Drake, 2011) to collect and analyse data. We exported some 'graphml' files to Gephi (Bastian, Heymann & Jacomy, 2009) for visualisation. The rest of this report is organised as follow. Section 2 describes data collection and processing. Section 3 delineates how communities are detected and then discusses the results. The last section concludes.

2. Dataset

2.1. Initial API Exploration

Our initial plan was to compare community detections among various social media platforms including GitHub, StackOverflow, LinkedIn, Kaggle, MeetUp, Reddit, and Twitter. GitHub and StackOverflow are most representative sources to our aims because users can reveal their preferences in R or Python through their projects on Git or interaction on question boards. LinkedIn is a professional network medium where users can specify their programming skills. Kaggle hosts data science competitions where competitors can run kernels on R or Python. MeetUp helps community creators organise events and talks, for example, running deep learning models with Python. MeetUp can be thought of as a bipartite graph. Reddit is a discussion board where users can post, answer, and share their comments about these languages tools. Twitter is a social microblogging site where users can follow and be followed by others who share similar (or opposing) content preferences.

Unfortunately, GitHub and StackOverflow API's (Application Program Interface) do not grant access to user profiles. LinkedIn offered some accesses but limited to personal accounts. Kaggle API allows users to download and submit competition predictions. Therefore, it is not useful. MeetUp only provides groups' but not users' information. Only Reddit and Twitter provide complete and open data for our analysis.

Following or being followed on Twitter cannot demonstrate one's actual preference nor "fluency" in either R or Python. For example, a product manager, who might have little background in data science, can follow influential Twitter users with expertise in R or Python. Having mentioned such caveat, we assumed users from our data were interested in data science and knew how to use at least one of these languages. The subsequent subsections discuss how we scrapped and processed the data via Twitter API.

2.2. Twitter Data Scrapping and Processing

Via tweepy (Roesslein, 2009), we started scraping data from the following Twitter accounts:

- R_programming (R & Py Tips),
- DataCamp,
- kaggle,
- rstudio,
- pycharm,
- RProgramming, and
- Pydatasci.

The data included users they followed and their followers. We chose the list above because we guessed them to be ego-centric nodes among data science communities on Twitter. Despite its name, 'R_programming' tweets and retweets R and Python tips for data analysis. 'DataCamp' is an online course site which has been offering introductory and intermediate-level data science courses with R and Python. 'Kaggle' attracts data science practitioners who need to run their scripts in Python and R. We suspected these accounts would have many users from both R and Python spheres. 'rstudio' and 'pycharm' are popular integrated development environments for R and Python respectively. RProgramming (not to be confused with 'R_programming') and 'Pydatasci' were randomly selected from Twitter search. We coined these accounts as "starting nodes".

Using the "starting nodes", we collected their full list of followers (first-level nodes) and followers' followers (second-level nodes) to build a larger and more diverse network. To limit the exponential increase in data size, we randomly sampled 50 to 100 second-level nodes. Using networkx (Hagberg, Swart & Chult, 2008), we saved each node's twitter as a graphml file. Finally, we stacked all graphml files into one single file named 'all_nodes.graphml'. This file was later used for further analysis. Figure 1A illustrates how data scrapping and processing were implemented. Due to tweepy API's limitation, we wrote four separate python scripts to collect data in a "divide-and-conquer" manner. Data scrapping was time-consuming since we downloaded data from 29 April to 9 May 2018.

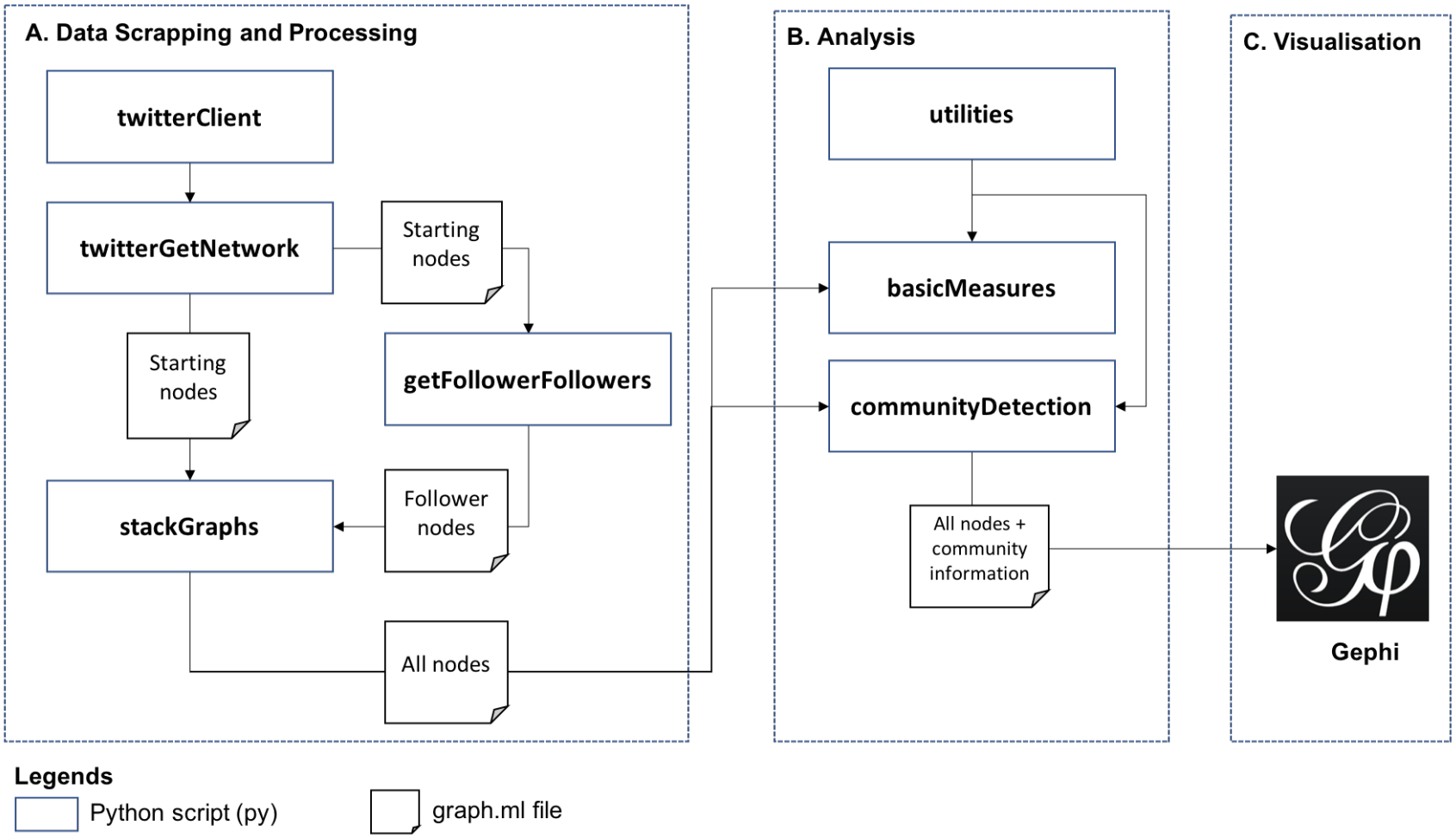

Figure 1. Flow of Project. Figure 1A shows how to scrap Twitter data. Figure 1B shows how graph files generated from data processing and then used for further analysis. The arrow represents the dependency among python files. For example, in Figure 1B, basicMeasures.py imports some handy functions from utilities to simplify analysis. Figure 1C shows that the refined version of "all nodes" graphml file is exported to Gephi.

2.3. Summary Statistics

The final graph file contains 48869 nodes (Twitter users) and 56098 edges. We developed a script named 'basicMeasure' (see Figure 1B) to obtain graph measures. The graph was weakly connected and had the following statistics:

- An average clustering coefficient of 0.043597,
- Transitivity of 0.000167, and
- Average shortest path length of 0.5583,

This indicates the graph was not dense. Therefore, we were not able to compute the number of connected components, diameter, and eccentricity. The biased sampling might cause this during the data collection.

We relied on degree and eigenvector centrality measures to obtain top 10 most central Twitter users but excluded Katz centrality which did not run successfully with networkx due to the large size of the graph. The most central nodes are reported in Table 1.

Table 1: Top 10 most central nodes by degree and eigenvector centrality measures

By degree centrality measure		By eigenvector centrality measure	
Twitter account	Central value	Twitter account	Central value
R_programming	0.130682985	RProgramming	0.178043042
DataCamp	0.111729434	rstudio	0.155952464
kaggle	0.106944844	hadleywickham	0.152827955
rstudio	0.105651156	RLangTip	0.146601788
pycharm	0.10314592	kaggle	0.131822114
pydatasci	0.06146043	pydatasci	0.131511072
RProgramming	0.018358043	DataCamp	0.125122921
TRUserGroup	0.006858598	TRUserGroup	0.123969272
Aveekchatterjee	0.003757855	Rbloggers	0.119762353
anacondainc	0.002197215	RConsortium	0.111502071

With the normalised degree centrality measures, our starting nodes were 7 out of top 10 most central nodes. We expected this result since our starting nodes had their most of followers. To solve bias, we switched to the eigenvector centrality measures. An advantage of the eigenvector centrality is that it incorporates the importance of the central nodes. As a result, except Kaggle,

pydatasci and DataCamp, top eigenvector-central nodes were R-related users, including hadleywickham (Hadley Wickham) who authors and develops the award-receiving and prominent "ggplot2" package to implement visualisation in R via grammar of graphics. Such results led us to wonder if R-related Twitter users are more influential compared to their Python counterpart in the data science community.

3. Analysis and Discussion

There are two types of community detection algorithms: member-based and group-based. Member-based community detection identifies nodes with similar node characteristics as a community. For example, nodes with similar degrees are in one community; nodes that are close are in one community. An example is Clique Percolation Method (Zafarani, Abbasi & Liu, 2014).

Group-based community detection identifies nodes as a community if they have specific group properties. For example, higher modularity than theoretical (when nodes are more densely connected within a community, compared to how connected they would be in an appropriately defined random network), for instance, Louvain's Method (Zafarani, Abbasi & Liu, 2014). In this project, we applied Clique Percolation Method (CPM) and Louvain's Method to identify communities.

3.1. Clique Percolation Method (CPM)

CPM uses cliques of a small size as seeds to find larger communities. Given an initial clique's size k, CPM first finds all cliques of such size in the network. Two cliques are linked if they share $k - 1$ nodes. Second, CPM replaces each clique with a node. If two cliques are linked, their node representations are also linked. Now each linked component in the node representation graph is a community.

3.2. Louvain's Method

Louvain's Method is an iterative process of two phases. The algorithm starts with all nodes being their communities. In the first phase, each node is removed from its current community to form a new community with one of its neighbour community to achieve a maximum modularity

gain. If no gain is possible, the node will not be removed from its current community. This process is repeated until one pass in which no node changes its community.

In the second phase, the community structure generated in the first phase is aggregated and then simplified. Each community in the first phase becomes a node in the second phase. The edge weight between two nodes in the second phase is the sum of lower-level edge weights between two communities. Each node in the second phase has a self-loop. For each such nodes, two times the number of edges, within its underlying Phase One community, is the value of its self-loop in the second phase.

These two phases are applied to the network repeatedly until one iteration in which the first phase does not make any change. One advantage of Louvain's Method is that the result of each "second phase" demonstrates a hierarchical level of communities.

3.3. Results

Using CPM, when we set k to 3, we identified 19 communities. When we increased k to 4 and 5, we identified three communities. Using CPM, we unwittingly removed more than 90 % of nodes resulting in small communities. As a consequence, we could not draw any robust inferences such as identifying overlapping nodes.

Due to the stochastic nature of Louvain's Method, we ran on a fixed random seed in Python. We detected 81 groups (see Figure 2). The detected groups were more interpretable, and the top five groups accounted for 45 % of the nodes. However, we could not evaluate these communities since we did not have the ground of truths. To work around, we verified if the top 10 most central nodes and our starting nodes were contained in the detected communities and conducted some data exploratory analysis in Gephi.

As illustrated in Figure 3, we discovered that the top 5 Louvain's communities contained 'Pycharm', 'Kaggle', 'DataCamp', 'RStudio' and 'R_Programming'. As expected 'PyCharm' was weakly linked to 'RStudio'. This results somehow vindicated that PyCharm followers might be more computer-science oriented. Compared to 'DataCamp', 'Kaggle' appeared to have fewer edges with 'PyCharm', 'RStudio', and 'R_Programming' although 'DataCamp' and 'Kaggle' were more connected. It was plausible that users who followed 'DataCamp' were aware of Kaggle Competition. Kaggle seemed to have a similar proportion of edges from 'PyCharm' and 'RStudio', but it was not clear for 'DataCamp'.

Meanwhile, DataCamp, RStudio, and R_Programming were more clustered together. Recall that 'R_programming' tweets and shares R and Python Tips. It might suggest that more R users are switching to Python for data analysis. Using the filter function in Gephi's data laboratory with keywords such as "stat" and "biostatistics", we could not find enough evidence if the Louvain's community, which contained 'R_studio', also included other statistics-related groups or Twitter accounts. Surprisingly, most of them were local or regional "chapters" or societies, for examples, 'RLadies_LP' and 'RPubsPrecent'.

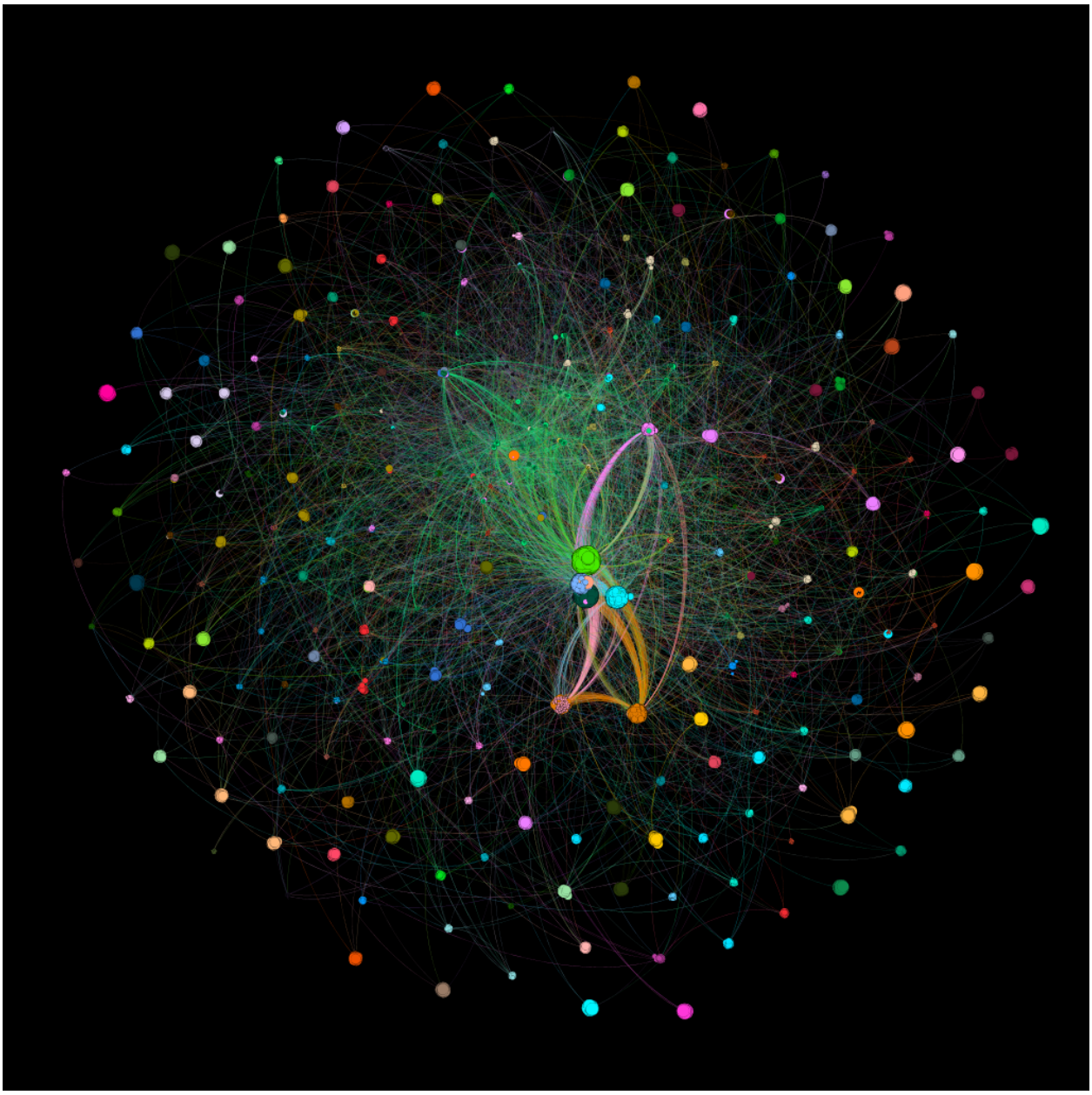

Figure 2: 81 communities detected by Louvain's methoid

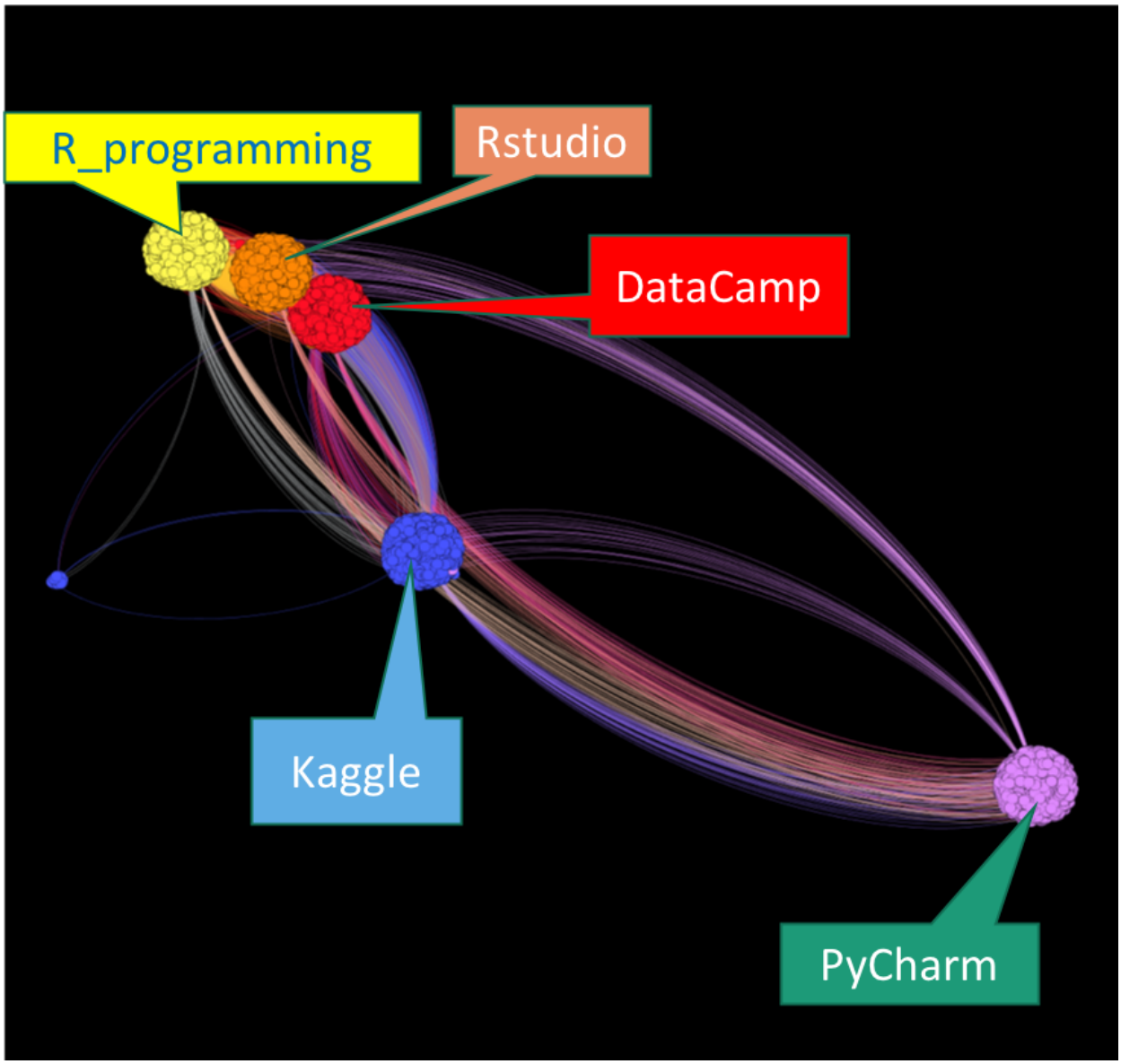

Figure 3: Top 5 Louvain's communities

4. Conclusion

In a nutshell, Louvain's method yielded more interpretable results compared to CPM although we were not able to identify the overlaps. We found that Kaggle attracted users from R and Python. However, there was no clear community demonstrated "bilingualism" in R and Python. Through data exploratory analysis, we vindicated that Python users (or followers) are more computer-science oriented, but we could not conclude R users have more background in statistics. We conjectured that there might be a growing number of R users learning or switching to Python. This project has a caveat. That is, following the "prominent" Twitters account does not mean that users are proficient in R or Python. We proposed to extend the concept of our project to other social media such as LinkedIn should we have the premium access. It can be a prototype for a skill-based collaborative project management or hiring syste

References

Ascher, D., & Lutz, M. (1999). *Learning Python*. O'Reilly.

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: an open source software for exploring and manipulating networks. *Icwsm*, *8*, 361-362.

Hagberg, A., Swart, P., & S Chult, D. (2008). *Exploring network structure, dynamics, and function using NetworkX* (No. LA-UR-08-05495; LA-UR-08-5495). Los Alamos National Lab.(LANL), Los Alamos, NM (United States)., Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

Gardener, M. (2012). *Beginning R: The statistical programming language*. John Wiley & Sons.

Matloff, N. (2013). *The art of R programming: A tour of statistical software design*. San Francisco: No Starch Press.

Piatetsky, G. (2018). R, Python Duel As Top Analytics, Data Science software – KDnuggets 2016 Software Poll Results. Retrieved from https://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html

Roesslein, J. (2009). tweepy Documentation. *Online] http://tweepy. readthedocs. io/en/v3*, *5*.

R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Summerfield, M. (2013). *Programming in Python 3: A complete introduction to the Python language*. Upper Saddle River, NY: Addison-Wesley.

Theuwissen, M. (2018). R vs Python for Data Science: The Winner is …. Retrieved from https://www.kdnuggets.com/2015/05/r-vs-python-data-science.html

Van Rossum, G., & Drake, F. L. (2011). *The python language reference manual*. Network Theory Ltd..

Zafarani, R., Abbasi, M., & Liu, H. (2014). *Social media mining*.