Stack Owerflow modeling and analyzing

Aliyev Gurban
g.aliyev@studenti.unipi.it
Student ID: 589206

Gaydamaka Anna
a.gaydamaka@studenti.unipi.it
Student ID: 589768

Parshina Kseniia
k.parshina@studenti.unipi.it
Student ID: 589769

ABSTRACT

Focusing on hashtags of publications of the well-known Stack Overflow website, an undirected graph was contructed as a tool for presenting connections between publications.

[1]

KEYWORDS

Hashtag, analyze, graph

ACM Reference Format:

Aliyev Gurban, Gaydamaka Anna and Parshina Kseniia. 2019. Stack Owerflow modeling and analyzing. In *Social Network Analysis '19.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Stack Overflow website is one of our most used resources. It was interesting for us to study what characteristics have users and publications, to find some connections. This structure can be well presented in the form of a graph (because the publications are linked by hashtags), which is demonstrated in further work.

[1] **Project Repositories**

Data Collection: https://github.com/sna-unipi/data-collection-blablacar
Analytical Tasks: https://github.com/sna-unipi/analytical-tasks-blablacar
Report: https://github.com/sna-unipi/project-report-blablacar

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$0.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 DATA COLLECTION

Data collection was the first and most challenging part of the project. The main problem was limits on crawling operations put by websites.

Selected Data Sources

We decided to use data of Stack Exchange, which is a useful online portal for programmers. We crawled 13308 records, however, due to computer limitations, only 10001 were used. Each record has 15 attributes. Some of the attributes are related to the questions while others belonged to the users asking the question. Four attributes were tags of the given question. Other attributes included user id, username, accept rate of the questions asked by the user, creation date of the question, etc. To create a graph the following idea was used: two publications are connected if they contain two or more the same hashtags.

Crawling Methodology and Assumptions.

Stack Exchange data was collected using the API key provided by the website. Also, we used a Python wrapper for crawling data, which is called StackAPI.

3 NETWORK CHARACTERIZATION

In this chapter of the report the constructed graph called g is discribed and compared with two structurally different networks: Erdős–Rényi model, ER, which represents a random model (1), and Barabási–Albert model, BA, being a preferential attachment graph (2). To construct them the same amount of nodes as the graph g has and formulas (1) – (2) are used. The following characteristics are obtained as a criteria:

Table 3.1. Comparison of graph g with ER and BA

Characteristics	Graph g	ER graph	BA graph
Nodes, N	10001	10001	10001
Edges, E	109426	109182	99910
Diameter	16	5	5
Average shortest path length	5.34776	3.29957	3.05988
Density	0.00219	0.00218	0.00199
Average clustering	0.48076	0.00229	0.01134
Closeness Centrality	0.1725 5358	0.32891 4650	0.47924 10
Betweennest Centrality of nodes	0.02452 5358	0.00076 4650	0.03953 10
Betweennest Centrality of edges	0.00611 (3170, 6336)	$5.4*10^{-5}$ (1927, 6927)	0.00084 (11, 13)

The table was obtained by Network X library for Python 3.

(1) $p = \frac{2*|E|}{|N|*(|N|-1)}$, where p is a probability of a node N_i having an edge E_j;

(2) $|E| = m * |N|$, is a network, whose degree distribution follows a power law.

For deeper evaluation we need to provide the following analysis.

Degree distribution

The node degree is the number of connections that it has with other vertices of the network. Therefore, the study of distribution is the first step to understand some common characteristics of a network typology. The nodes of the most real networks have a different number of degrees, which is due to the presence of hubs - nodes, characterized by a degree that is clearly higher than the average, and therefore defined as central nodes. This effect is clearly presented by graph g on the Fig. 3.1.

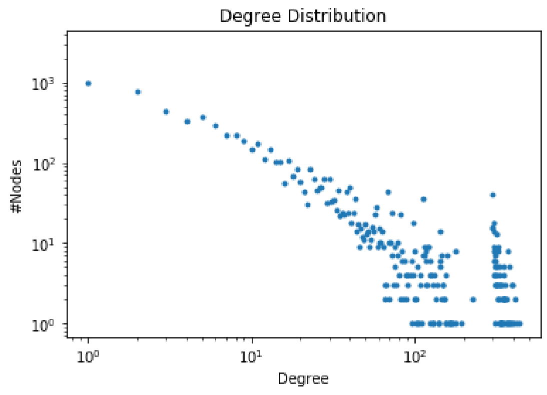


Figure 3.1. Degree of distribution of graph g

The most important hub in the network topology, with a grade equals 433. It represents hashtags c#, .net, asp.net, vb.net, clr has the highest degree 433. The graph includes 3029 nodes with degree 0, so they represent separated components themselves.

The degree distribution illustrates us that the network has hubs and that mostly nodes have degree less than 5.

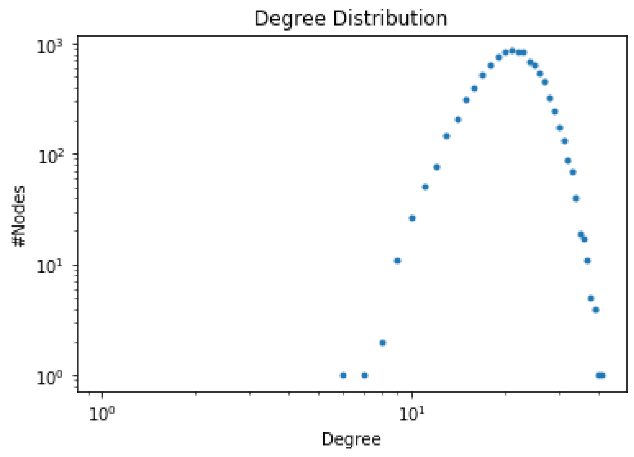


Figure 3.2. Degree distribution of ER graph

In random networks the emergence of hubs is not possible since the degree of each node will tend to be similar to others. The distribution trend will approximate a curve Gaussian (Fig. 3.2). At the same time, the distribution pattern in real networks is best described by a Zipfian curve (Fig. 3.3).

The dataset contains 5 hubs which degree is higher than 400, ER graph doesn't contain any hub and BA graph contains 8 nodes with degree more than 400.

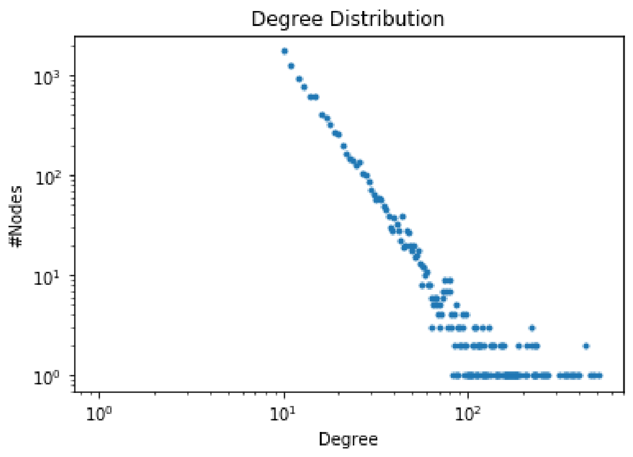
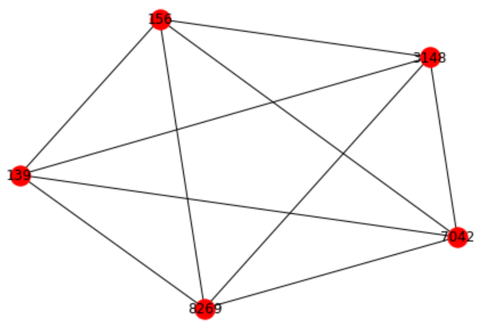


Figure 3.3. Degree distribution of BA graph

At the same time, as we see in Table 3.1, density of the all three graphs is quite similar.

Density

The local clustering coefficient measures the local density of links in a node's vicinity. Density of the graph g is around 0.002.



.

Figure 3.4. Triangles of node 139 of graph g

For example, node 139 has 6 triangles. Such metric is necessary for constructing a clustering coefficient. For the node 139 clustering coefficients equals 1 while the average value is around 0.48.

For the ER graph with the lowest value of average clustering, we visualized neigbours of the node 1 to provide an idea that if a node doesn't have any 'triangle', that stands for a lack of a strong 'relationship', the Clustering Coefficient (1) equals 0 (Fig. 3.5).

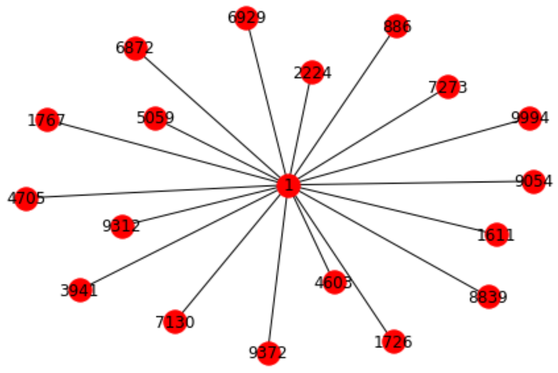


Figure 3.5. Triangles of node 1 of ER graph

Connected components

Number of connected components of g is 3436. The size of the biggest one is 5839, while all the next components include significantly less nodes, for example, the second biggest includes 14 nodes. The graph also includes 3029 components with a unique node.

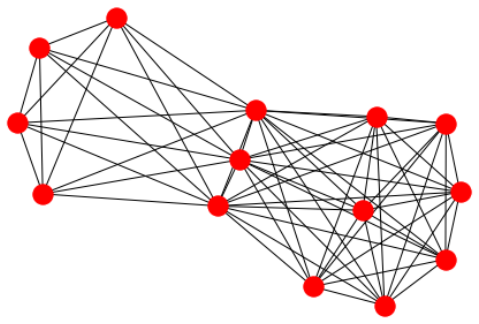


Figure 3.6. Second biggest component, 139, of graph g

That is why the graph g is called a disconnected graph.

Path

The graph g is an incomplete and weakly connected then we couldn't count the shortest paths between all the nodes, therefore, we explored the biggest component. The longest shortest path, called diameter, is 16. The graph doesn't contain a Hamiltonian path. The average shortest path length is 5.34776, which is, comparing to ER and BA graphs, is significantly bigger number (Table 3.1). Furthermore, the use of path and shortest path analysis is fundamental in assessing the centrality of nodes and edges using measures such as centrality and betweenness.

Centrality

Centrality measures could be usefull in the following sence: they are low for node pairs in the same community and high for nodes that belong to different communities. Such information is applicable for Community Discovery. In the table 3.1 the highest values and nodes of them are provided for centrality measures.

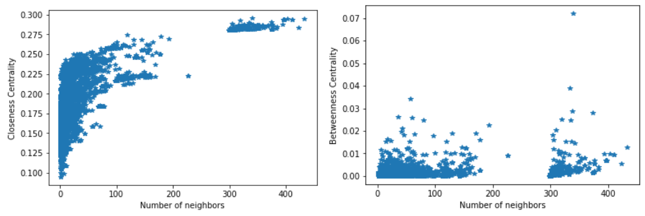


Figure 3.7. Centrality measures distributions of graph g

The node with the biggest amount of neighbors has high Closeness Centrality, in this case the value for the node 9935 of graph g is around 0.17222, while the highest value 0.17256 is reached by the node 5358 as well as it has the highest value 0.02452 in Betweenness Centrality. A high value in Betweenness Centrality means that a node has big number of paths that come using it, we could assume that such node is a hub or/and has many neighbors. Our assumptions are true, because this node contains a hashtag "c#" which is the most popular one.

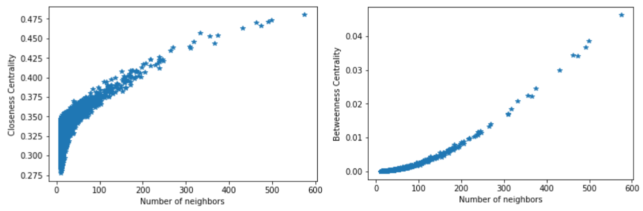


Figure 3.8. Centrality measures distributions of BA graph

Degree Centrality is one of the basic measures of the centrality of a node in the network. This is based on the assumption that the importance of a node is proportional to the number of its neighbors. While in a BA model hubs are presented by the most central nodes.

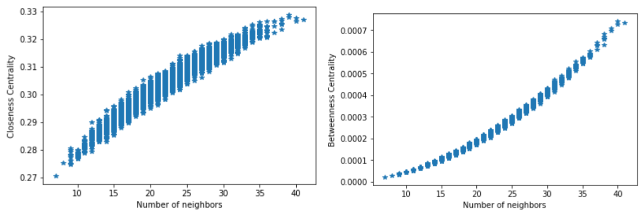


Figure 3.9. Centrality measures distributions of ER graph

Analyzing graphics 3.7-3.9 we could sum up that distribution of centrality measures differs for presented graphs. As can be seen clearly, centrality measures are related on the number of neighbors of a node: the bigger amount of neighbors is, the higher its metric. As for the ER graph , this is due to the probabilistic creation of an edge between nodes.

4 TASK I: LINK PREDICTION

Considering only the graph g link prediction was made to grasp how a network evolves. For an application of unsupervised approaches such as Common Neighbors, Jaccard , Adamic Adar and KATZ [1] we devided the network in a training set (80% of the edges) and a test set (20% of the edges) randomly. The Table 4.1 with final accuracy values is presented below:

Table 4.1. Accuracy of predictions

Approach	Accuracy (*n*)
Common Neighbours	74.95 (16510/87730)
Jaccard	82.88 (18257/87730)
Adamic Adar	81.32 (17913/87730)
Katz	71.70 (15794/87730)
SimRank	47.80 (10419/87694)

Neighborhood measures :

Common Neighbours. Tagline: *The more friends we share, the more likely we will become friends.*

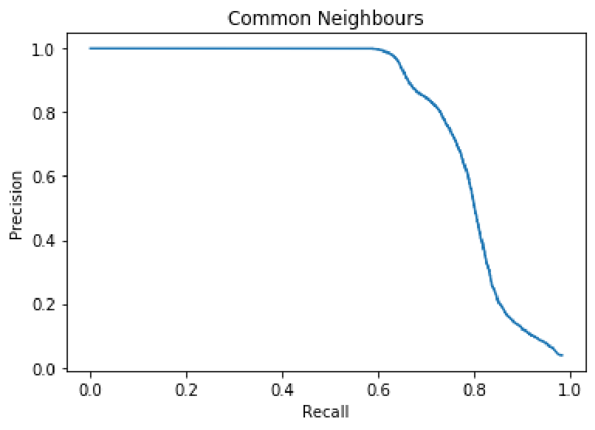


Figure 4.1. Visualization of metrics for Common Neighbours

Jaccard. Tagline: *The more similar our friends circles are, the more likely we will become friends.*

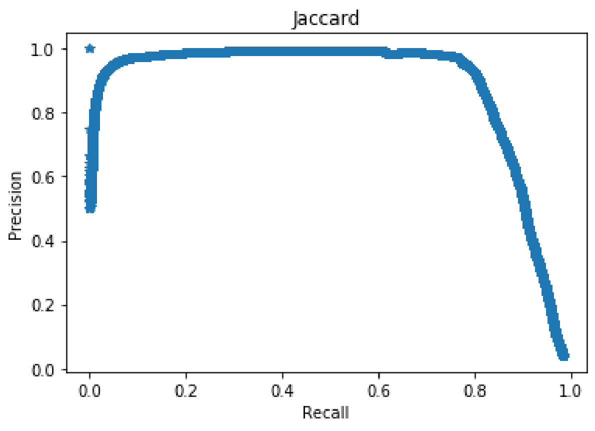


Figure 4.2. Visualization of precision and recall metrics for Jaccard method

Adamic Adar. Tagline: *The more selective our mutual friends are, the more likely we will become friends.*

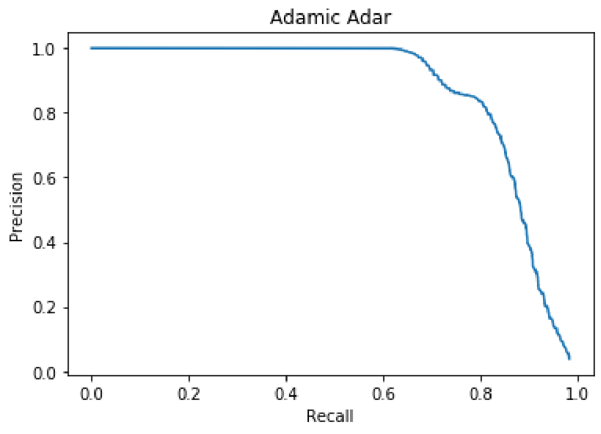


Figure 4.3. Visualization of precision and recall metrics for Adamic Adar algorithm

Path-based measures: KATZ. It computes the weighted sum over all the paths between two nodes.

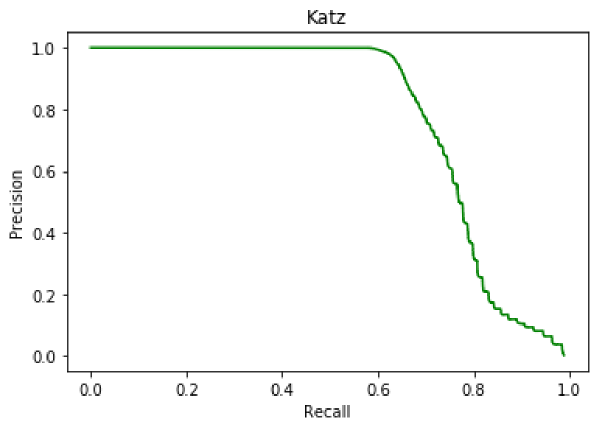


Figure 4.4. Visualization of precision and recall metrics for Katz method

Ranking

SimRank. Tagline: *two nodes are similar to the extent that their neighborhoods are similar*.

After calculating the predicted links with each of the different approaches, only the n edges with the highest score were taken into account, where n is equal to the number of links initially removed from the graph g (20%). This is hold due to the nature of Link Prediction algorithms which assign a score to all the possible edges which are not yet present in the graph. According to the algorithm, those with the highest score will appear more likely. According to the Table 4.1, Jaccard method gave the best results comparing with others.

Other possible algorithms [1] are not presented because of computer's restrictions as well as there is no SimRank visualization, for which we used the graph g without isolated nodes. We could sum up that the shape of the curves is similar: between values 0.6-0.8 the first recession is appeared. Different from others, Jaccard algorithm curve increases before 0.1 and decreases after 0.8, presenting the biggest square under itself.

For Link Prediction task we could conclude that mostly edges, to which algorithms assigned a higher score, were predicted correctly.

TASK II: COMMUNITY DISCOVERY

Relying only on a high-level analysis of the network can lead to the generalization problem. This problem results in missing useful insights of the complex network structures which could be used in studying network models and their phenomena. To solve generalization problem, community discovery (community detection) was introduced in our analysis. Community discovery uses algorithms to identify meso-scale communities in the network structure. We will focus on four algorithms listed below to identify communities:

1. K-cliques
2. Label Propagation
3. Louvain
4. Demon

Each of these algorithms executes partition based on a different property of nodes. Working mechanism of each algorithm used and results will be reported. We also considered that as each algorithm detect communities on a different property leading to different partitions. Such a characteristic of community discovery is called an ill posed problem of community discovery. We tried to pass over this obstacle by evaluating algorithms and comparing them to one another.

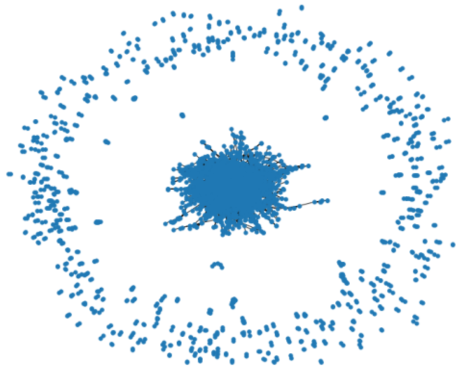


Figure 4.5. Network structure

To decrease difficulty level of community discovery, it was assumed that our network is static, meaning communities do not change during the time. Then, we plotted our network (Fig. 4.5) to see the structure of the network obtained. The graph showed us that there are some nodes which do not have any edge or have few edges. We assumed that those nodes are noise and they prevent us to see a better picture of partition while running algorithms. Therefore, we did not plot communities having few nodes.

K-clique

K-clique belongs to algorithms using structure definition as a partition property. This algorithm extracts complete graphs of size k as communities, meaning existence of non-covered nodes is possible. Another name of this algorithm is c-percolation, meaning this is a bottom-up approach. We ran k-clique algorithm on CDlib package installed on Python, setting k value of 8. We used optimization function to choose the value of k. Our optimization function is based on ER modularity, and the function reached its maximum at k=8. The coverage of the network was also not so low – 49%. So, we continued our analysis setting 8 as the value of k. Also, consider that k-clique algorithm creates overlapping communities (in our case, too), meaning a node can belong to several communities. Visualization of partition is shown in Fig. 4.6. The graph is shown without overlaps to simplify the view. Moreover, some insights for evaluation of k-clique partition are reported in Table 4.2.

Table 4.2. K-clique partition

K-Clique (k=4)	min	max	score	st/dev
Internal Density	0.035	0.25	0.244	0.025
Average Degree	7	130.422	12.456	12.062
Conductance	0	0.973	0.548	0.290
Modularity (ER)	-	-	0.914	-

While algorithm identified 226 communities, we can observe 6 big communities from the graph. One of those communities (coloured red) looks bigger, which makes us reason that stakoverflow users are strongly connected to one another according to k-cliques algorithm. Average degree of 12 can also support our assumption. On the other hand, we can imply that other communities obtained are not so representative, which is not a good sign.

Necessary details for evaluation of the algorithm in

Table 4.2 are conductance and modularity. These variables are indicators of the partition quality and range between [0; 1]. Conductance of 0.548 informs us about moderately good quality of partition while modularity of 0.914 tells that partition was well-executed.

Label Propagation

Label Propagation algorithm is also using percolation as a partition property and is a bottom-up approach. In this algorithm, there are 3 steps: 1) in the first iteration, each node randomly changes its label to the label of one neighbor; 2) at each subsequent iteration, every node accepts the label of majority of neighbors; 3) iterations continue till consensus reached. Partition of label propagation does not lead to overlaps and covers all nodes of the network. There are 859 communities indentifies, and main communities are colored in Fig. 4.7 while results of fitness functions for clustering evaluation are shown in Table 4.3.

Table 4.3. Label Propagation partition

Label Prop.	min	max	score	st/dev
Internal Density	0.019	0.25	0.217	0.055
Average Degree	1	94.812	3.103	5.7
Conductance	0	0.857	0.23	0.253
Modularity (ER)	-	-	0.534	-

The graph has the same number of main communities, with comparable sizes. However, the graph shows that clusters lost their strong connectivity. Lower average degree approves our observation. Also, average level of modularity supports our idea. However, conductance became much lower with Label Propagation algorithm, which is against our assumption.

Louvain

Louvain algorithm uses internal density as a clustering criterion. The algorithm works as follows: a function is defined for density (e.g. modularity), and algorithm finds communities given maximum possible value of the function. This algorithm covers all nodes and does not have overlaps. Using optimization function (which was previously used for k-clique), we set randomization and resolution of 0.9. The algorithm under given parameters identified 454 communities. However, only 4 of them are clearly seen on Fig. 4.8. Moreover, values for internal evaluation of the partition are given on the table 4.4. In this algorithm, internal density and average degree are still low. However, conductance score of 0.014 and modularity score of 0.878 infer that algorithm could discover well separated communities based on internal density.

Table 4.4. Louvain partition

Louvain	min	max	score	st. dev.
Internal Density	0.006	0.25	0.225	0.058
Average Degree	1	225.044	3.041	11.302
Conductance	0	0.429	0.014	0.058
Modularity (ER)	-	-	0.878	-

Demon

Demon is node-centric algorithm working with percolation and using bottom-up approach. This algorithm can lead to partial coverage of partition and overlaps, too. Also, its working principle includes extracting ego network of each node and performance of Label Propagation. In Demon algorithm, we used random search optimization function to parameters as grid search takes much more times. Choosing 5 random cases, we got epsilon of 2.3 and minimum community size of 6 giving maximum modularity. Given parameters, Demon algorithm identified 2057 communities. However, partition covered 65% of the network. We plotted visual representation of the network in Fig. 4.9 and shared results of the fitness functions in Table 4.5.

Table 4.5. Demon partition

Demon	min	max	score	st. dev.
Internal Density	0.081	0.25	0.221	0.034
Average Degree	3.429	299	44.274	75.215
Conductance	0	0.98	0.524	0.277
Modularity (ER)	-	-	78	-

In graph 2.5, we see only 4 coloured communities. When we look at Table 4.5, we see that internal did not significantly change in comparison to the previous algorithm. However, average degree abnormally increased. These numbers can imply communities have more connections to outside nodes and less connections to nodes inside. The reason can be large amount of communities. Conductance is on average, meaning partition fitted not so well. However, modularity is 78, which is abnormally high number (usually modularity is between 0 and 1). The reason of such a high number can be that nodes with high degree have many edges connected outside their communities. Also, the reason of low internal density and higher average degree can be explained with the fact that each community has few nodes, each node having very high degree.

After conduction of internal evaluation, we analysed Label Propagation partition by looking at distribution of languages in communities 0 and 1. In community 0, the main tags used are c# and .net while in community 1 the main tags were sql-server and sql.

<u>External Evaluation</u>

After running algorithms and conducting internal evaluation with fitness functions, we compared algorithms to each other using scores of fitness functions, normalized mutual information (NMI) and Normalized F1 (NF1). Also, we compared algorithms by planting network structures with ground truth partition.

While comparing partition on internal density, we saw that all algorithms used above have similar scores. Demon algorithm had the highest score in average degree, while Louvain had the best conductance score. When it comes to modularity, Demon algorithm had abnormally high score (78) while had the the second highest result which was close to 1 (0.878).

Also, we plotted graphs of fitness functions scores with respect to community size (graphs are in python notebook). It was found out that Label Propagation, Louvain and K-clique have decreasing internal density when community size increases. However, Demon communities have closer community sizes and internal density scores. When it comes to average internal degree distribution, all algorithms had similar shapes (Louvain's graph had a bit different shape). In comparison of conductance distribution, it was observed that Label Propagation had lower score for a given size of community in comparison to DEMON and K-clique graphs, and higher scores in comparison to Louvain.

NMI can be used under two conditions: 1) compared partitions include the same set of nodes, 2) each of the compared partitions is not overlapping. So, we used NF1 score for algorithms which did not satisfy those conditions. Using NMI, we compared Louvain and Label Propagation algorithms and obtained resemblance score of 0.77.

Also, we calculated NF1 resemblance scores and shared results in Table 4.6.

Table 4.6. Resemblance scores

NF1	K-clique	L. Propag.	Louvain	Demon
K-clique	-	0.045894	0.001311	1.061332
L. Propag.	-	-	0.279465	0.038903
Louvain	-	-	-	0.000114
Demon	-	-	-	-

From the table above it is seen that K-clique and Demon algorithms have abnormally high resemblance of 1.06. Consider that resemblance between Louvain and Label Propagation is lower in comparison to NMI score. However, this is the second highest NF1 score.

The last external evaluation method we used was testing algorithms against synthetic graphs. The generator used was LFR producing non overlapping and complete ground truth partition. In addition, test results were given both with NMI and NF1 scores. Results depended on some parameters, the main parameter to check was the the fraction of intra-community edges incident to each node (mu). For mu = 0.1, NMI score both of Label Propagation and Louvain algorithms was 0.599. When it comes to NF1 scores, all algorithms, in surprise, had the same result: 0.017514. It means algorithms have the same effect in low mu. So, we checked results for mu values of 0.5 and 0.9. However, networkx functions returned equal results, again.

To check results of networkX, we built Fig 4.10 with cdlib. This graph helps us to compare Louvain and LP algorithms for different mu levels (0.1 for g1, 0.3 for g2, 0.5 for g3, 0.7 for g4, 0.9 for g5). Algorithms were compared based on NMI score. In this case, we see that algorithms have equal scores in high mu levels, vice versa. We decided these ambiguous results are because of randomness of synthetic network creation.

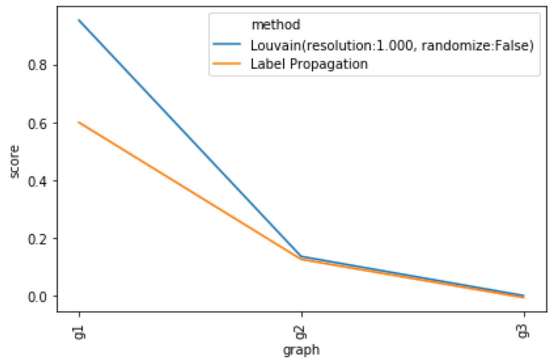


Figure 4.10. Comparison against synthetic network

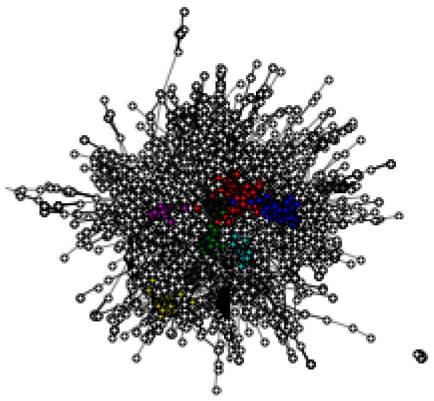


Figure 4.6. K-clique partition (k=8)

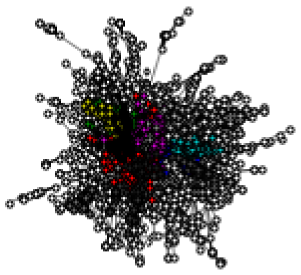


Figure 4.7. Label Propagation partition

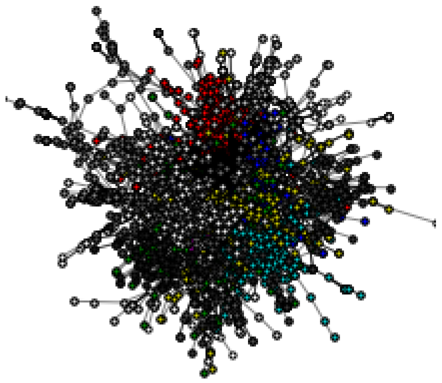


Figure 4.8. Louvain partition

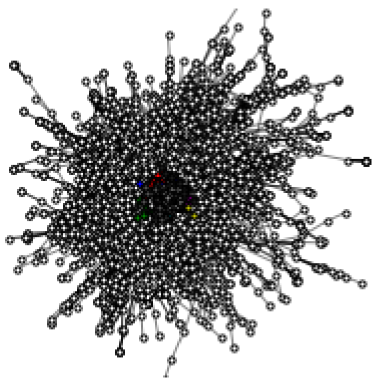


Figure 4.9. Demon partition

5 Open question

As an open question we decided to consider the following problem: *Will the user necessarily be popular if he uses popular hashtags?* Or, in other words, is there a connection (correlation) between such parameters as reputation, acceptance rate (the percentage of answers accepted based on the questions asked by the user), views (of the publication, made by the user) and hashtag popularity?

Our first task was to find out which hashtags are popular. The hashtag popularity was determined by the degree of the node. The maximum degree is 433. We set a threshold value of 400 degree. All hashtags tied to nodes that have a degree of 400 or more are considered popular. Five nodes satisfied the described above threshold. Some hashtags (c#, .net, asp.net) belonged to almost each of these five publication. After removing duplicates, the following list of popular hashtags was obtained: c#, .net, wpf, winforms, asp.net, webforms, rest, webrequest anonymous-access, vb.net, clr.

On the next step we found how many popular hashtags has each publication. After that, it became possible to find numerical correlations between the number of popular hashtags (in the table - count) and reputation, views, acceptance rate (ar). Also a correlation matrix was built (Fig. 5.1).

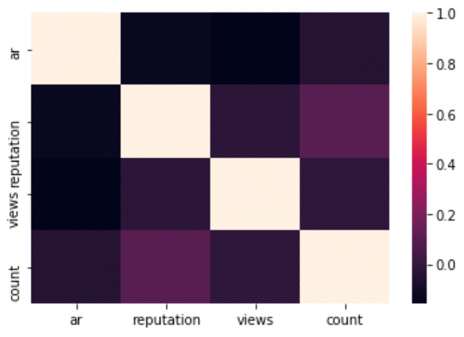


Figure 5.1. Correlation matrix between attributes

As can be seen from the correlation matrix, the correlation values are very low. We were interested in pairs count-ar, count-reputation, count-views. The highest correlation is given by the pair count-reputation (0.103814), and the lowest - count-views (-0.014001). Thus, we can conclude that the popularity of the user is weakly dependent on how popular hashtags he uses.

This result can be explained by the fact that the graph is constructed in such a way that only if there are at least two identical hashtags, the publications are linked to each other. Thus, it is possible that the user used only one popular hashtag, so the publication was not connected to the others, but the user has, for instance, a high reputation rate. For instance, hashtags .net, escaping, string.format have only one co-opearance, but the owner of publication, Jeff Atwood, has the highest reputation rate = 100.

6 Conclusion

In conclusion, the study of the network has certainly led to interesting analyses, above all allowing the study of popular programming hashtags all over the world. A further step forward for a more in-depth and meaningful analysis of the network can be obtained by addition of other attributes of publications and/or of users. Moreover, it could lead to prediction of popular topics and evaluation of users' activities.

Sourses:

1. David Liben-Nowell, Jon M. Kleinberg: The link prediction problem for social networks. CIKM 2003
2. Albert-László Barabási: Network Science.
3. StackAPI. https://stackapi.readthedocs.io/en/latest/