NETWORK ANALYSIS OF THE STACK OVERFLOW TAGS

Zahir Edrees

Department of Computer Engineering, Faculty of Engineering, Karabuk University, Karabuk, Turkey
zahir-mohamed2010@hotmail.com

KEY WORDS: Network Density, Modularity, Betweenness Centrality, Closeness Centrality, Node Degree, Degree Distribution.

ABSTRACT:

In this paper we made analysis for the stack overflow tags by using different criteria's in network science, one of the advantages of network analysis is that complex of connections can be made cleared, we started this work in first step by extracted data from dataset after that applied network concepts node degree distribution, node importance (centrality measures), also we provided a brief demonstration of how we can use graph network and tools to analyze semi-structured text as (Tags).

1. INTRODUCTION

Stack Overflow is the largest online community for programmers to learn, share their knowledge, and advance their careers. Stack Overflow Dataset consists of Posts, Post Links, Tags, Users, Votes, Batches and Comments. We want to make analysis to this data in specific time between (2018 and 2019).

Question tags are an important part of all submitted questions in Stack Overflow, because it allowing answering users to monitor questions relevant to their old of expertise and to answer promptly to submitted questions. Only users holding an "advanced" status in Stack Overflow are allowed to generate new tags, with the rest of the community limiting themselves in using the existing tags for their questions. Tagging the submitted questions is mandatory (i.e., there is a minimum of one tag per question), and each question can contain up to five tags.

We carry out an analysis of the Stack Overflow tags viewed as a network, or a graph. Specially, we aim to get some insight about the user communities by representing tags and their co-occurrences as a graph, where the graph nodes are the tags themselves, and an edge between two nodes exists if the corresponding tags are found together in the same Stack Overflow question.

The resulting network edges are weighted, the higher the weight of the relevant edge between them will be; for example, if the tag java is found to co-exist with the tag android in 1000 questions, the weight of the edge between the nodes java and android in our graph will be 1000.

This paper is organized as follows:
Section 2: describe briefly the data acquisition process and present some summary statistics of the data set.
Section 3: expose the construction of our first graph based on the whole data set, along with some limited analysis.
Section 4: describes the meaningful reduction of our raw data set.
Section 5: building the adjacency matrix.
Section 6: deals with various measures of node degree distribution.
Section 7: deals with various measures of node importance (centrality).
Section 8: conclusion and discussion.

2. DATA ACQUISITION AND SUMMARY STATISTICS

For data set extraction, we first created the project on google cloud after that we connected this project with google datasets and we used google big query tools to extract data Figure 1: shows the steps.

2.1 Dataset Overview

Stack Overflow Dataset consists of following files that are treated as tables in our Database Design Figure 2:

- Posts
- Post Links
- Tags
- Users
- Votes
- Batches
- Comments

The details of the raw data parsing in Table 1, the results are:

- A list of the used tags, ordered by decreasing frequency. The summary statistics of the raw data are shown in Table 1.
- Load("tag_freq")

\[
\text{tag_freq}[:\text{rel} : \text{freq} = \text{freq} / \text{sum(tag_freq.freq)} * 100]
\]

According to Table 2, the 10 most frequent tags account for about 21% of all tag occurrences the raw data. We can similarly explore the other end of the list that is how many tags appear very infrequently in the data set:

- Length (which (tag_freq.freq < 100))
  Sum: 38,726
- Length (which (tag_freq.freq < 10))
  Sum: 13,871

This contribution has been peer-reviewed.
https://doi.org/10.5194/isprs-archives-XLIV-4-W3-2020-195-2020 | © Authors 2020. CC BY 4.0 License.
### Table 1. Statistical of raw data, time between (2008 and 2019).

| # | Tag name | Count   | Relevant frequencies |
|---|----------|---------|----------------------|
| 1 | JavaScript | 1,955,557 | 3.48                 |
| 2 | Java     | 1,641,102 | 2.92                 |
| 3 | c#        | 1,385,220 | 2.46                 |
| 4 | Python   | 1,359,126 | 2.42                 |
| 5 | Php      | 1,335,050 | 2.37                 |
| 6 | Android  | 1,254,482 | 2.23                 |
| 7 | Jquery   | 978,412   | 1.74                 |
| 8 | Html     | 970,699   | 1.73                 |
| 9 | c++      | 656,969   | 1.17                 |
| 10 | Css      | 649,436   | 1.15                 |
| Total | | 12,186,053 | 21.65               |

### Table 2. Top 10 Tags time between (2008 and 2019).

| # | Tag name | Count   | Relevant frequencies |
|---|----------|---------|----------------------|
| 1 | JavaScript | 1,955,557 | 3.48                 |
| 2 | Java     | 1,641,102 | 2.92                 |
| 3 | c#        | 1,385,220 | 2.46                 |
| 4 | Python   | 1,359,126 | 2.42                 |
| 5 | Php      | 1,335,050 | 2.37                 |
| 6 | Android  | 1,254,482 | 2.23                 |
| 7 | Jquery   | 978,412   | 1.74                 |
| 8 | Html     | 970,699   | 1.73                 |
| 9 | c++      | 656,969   | 1.17                 |
| 10 |Css      | 649,436   | 1.15                 |
| Total | | 12,186,053 | 21.65               |

### Table 3. Statistical of raw data, time between (2018 and 2019).

| # | Tag name | Count   | Relevant frequencies |
|---|----------|---------|----------------------|
| 1 | JavaScript | 1,955,557 | 3.48                 |
| 2 | Java     | 1,641,102 | 2.92                 |
| 3 | c#        | 1,385,220 | 2.46                 |
| 4 | Python   | 1,359,126 | 2.42                 |
| 5 | Php      | 1,335,050 | 2.37                 |
| 6 | Android  | 1,254,482 | 2.23                 |
| 7 | Jquery   | 978,412   | 1.74                 |
| 8 | Html     | 970,699   | 1.73                 |
| 9 | c++      | 656,969   | 1.17                 |
| 10 |Css      | 649,436   | 1.15                 |
| Total | | 12,186,053 | 21.65               |

### 3. REDUCED GRAPH

We presented in Table 2 a full view for stack overflow dataset and we want to reduce these data to achieve more accuracy for our dataset, we will take data from dataset in (2018 and 2019) in addition to add new condition for tags, only tags that have occurrences more than 500 times. Table 3 shows dataset statistics after reduced. Figure 4, 5 shows charts for Top Tags and frequencies between (2018-2019).

| # | Tag name | Count   | Relevant frequencies |
|---|----------|---------|----------------------|
| 1 | JavaScript | 1,955,557 | 3.48                 |
| 2 | Java     | 1,641,102 | 2.92                 |
| 3 | c#        | 1,385,220 | 2.46                 |
| 4 | Python   | 1,359,126 | 2.42                 |
| 5 | Php      | 1,335,050 | 2.37                 |
| 6 | Android  | 1,254,482 | 2.23                 |
| 7 | Jquery   | 978,412   | 1.74                 |
| 8 | Html     | 970,699   | 1.73                 |
| 9 | c++      | 656,969   | 1.17                 |
| 10 |Css      | 649,436   | 1.15                 |
| Total | | 12,186,053 | 21.65               |
4. BUILDING THE ADJACENCY MATRIX

We used Big Query ML for building the adjacency matrix.

Table 4. Top 60 Tags and frequencies between (2018-2019)

| #  | Tag          | Questions | #  | Tag          | Questions | #  | Tag          | Questions |
|----|--------------|-----------|----|--------------|-----------|----|--------------|-----------|
| 1  | Python       | 446792    | 21 | Arrays       | 64712     | 41 | .net         | 31177     |
| 2  | Javascript   | 412459    | 22 | Json         | 60956     | 42 | Tensorflow   | 30673     |
| 3  | Java         | 287777    | 23 | excel        | 58213     | 43 | ruby-on-rails| 30637     |
| 4  | c#           | 216214    | 24 | django       | 57944     | 44 | asp.net      | 30563     |
| 5  | Android      | 209600    | 25 | laravel      | 56233     | 45 | Dataframe    | 29769     |
| 6  | Php          | 191120    | 26 | typescript   | 55939     | 46 | Azure        | 29511     |
| 7  | Html         | 162175    | 27 | sql-server   | 55642     | 47 | Linux        | 28455     |
| 8  | python-3.x   | 118880    | 28 | c            | 51434     | 48 | Flutter      | 28010     |
| 9  | Angular      | 111015    | 29 | vba          | 44804     | 49 | Ajax         | 27429     |
| 10 | Reactjs      | 110091    | 30 | spring-boot | 43358     | 50 | Database     | 26923     |
| 11 | Css          | 108146    | 31 | Firebase     | 43254     | 51 | Bash         | 25709     |
| 12 | R            | 106719    | 32 | react-native | 41505     | 52 | Oracle       | 25609     |
| 13 | node.js      | 105502    | 33 | Spring       | 40074     | 53 | xml          | 24652     |
| 14 | C++          | 102914    | 34 | Regex        | 38261     | 54 | numpy        | 24623     |
| 15 | Sql          | 101358    | 35 | Wordpress    | 37252     | 55 | kotlin       | 24550     |
| 16 | Jquery       | 98008     | 36 | Docker       | 36009     | 56 | string       | 24458     |
| 17 | Mysql        | 89424     | 37 | amazon-web-services | 35569 | 57 | git         | 24148     |
| 18 | Ios          | 82735     | 38 | vue.js       | 34020     | 58 | powershell   | 23231     |
| 19 | Swift        | 78654     | 39 | Mongodb      | 32929     | 59 | asp.net-mvc  | 22935     |
| 20 | Pandas       | 70671     | 40 | Postgresql   | 31441     | 60 | apache-spark | 22866     |

Figure 3. Bar chart for Tags and frequencies between (2018-2019)
3.1 Co-occurring Tags

We created a big table to find tag occurrences; after that, we can find tags that usually go together throw this query:

```
SELECT * FROM `seraphic-lock` 259921.StackOverflow.posts_questions_partitions n3 WHERE tag1 = 'javascript' ORDER BY percent DESC LIMIT 20
```

| Tag1       | Tag2       | Questions | Questions_tag1 | Percent  |
|------------|------------|-----------|----------------|----------|
| JavaScript | Javascript | 412459    | 412459         | 1        |
| JavaScript | Html       | 70779     | 412459         | 0.171603 |
| JavaScript | Jquery     | 62587     | 412459         | 0.151741 |
| JavaScript | Reactjs    | 46956     | 412459         | 0.113844 |
| JavaScript | node.js    | 36804     | 412459         | 0.089231 |
| JavaScript | Css        | 31007     | 412459         | 0.075176 |
| JavaScript | Arrays     | 18630     | 412459         | 0.045168 |
| JavaScript | Angular    | 18387     | 412459         | 0.044579 |
| JavaScript | Php        | 17742     | 412459         | 0.043015 |
| JavaScript | Typescript | 14799     | 412459         | 0.03588  |
| JavaScript | Vue.js     | 13846     | 412459         | 0.031572 |
| JavaScript | Json       | 12179     | 412459         | 0.029528 |
| JavaScript | React-native | 9182   | 412459         | 0.022262 |
| JavaScript | Ecmascript-6 | 8414   | 412459         | 0.0204   |
| JavaScript | Angularjs  | 8051      | 412459         | 0.01952  |
| JavaScript | Express    | 6853      | 412459         | 0.016615 |
| JavaScript | Firebase   | 6468      | 412459         | 0.015682 |
| JavaScript | Regex      | 5677      | 412459         | 0.013764 |

Table 5. ‘JavaScript’ tag occurrences with other tags

These groupings shows: ‘JavaScript’ is related to ‘html’, ‘jQuery’, etc. …

3.2 Building the Adjacency Matrix

We used Big Query ML for building adjacency matrix. Big Query ML enables users to create and execute machine-learning models in Big Query using standard SQL queries.

3.3 Clusters and Connected Components

First thing is to check network globally connected, the graph after checked it is not globally connected because some of nodes are isolated, in Table 7 the following nodes are part of isolated nodes. Figure. 7 shows graph nodes with their edges.

| # | Tag           |
|---|---------------|
| 1 | Jcarousellite |
| 2 | Gitorious     |
| 3 | Scrolloindex  |
| 4 | double-submit-prevention |
| 5 | Viewlayout    |
| 6 | Friendrequest |
| 7 | Markdowndeep  |
| 8 | user-agent-switcher |
| 9 | db.js         |
| 10| pytest-bdd    |

Table 6. Shows some isolated tags
We grouped our nodes with their edges to five groups see Figure 8, our reduced graph contains 47,972 tags but we just took 500 tags (valuables) that have the highest frequency degree to visualize the graph. Figure 8 shows these grouping for tags and some statistical information we will explain it in more details.

4. NODE DEGREES

The degree of a node is simply the number of its direct neighbours, or equivalently the number of the links in which the node is involved, Figure 9 shows some of our tag degrees and average degree for graph after grouping it to five groups.

Table 7. Top 20 Tags degrees

| # | Tag      | Degree |
|---|----------|--------|
| 1 | mysql    | 55     |
| 2 | python   | 51     |
| 3 | c#       | 49     |
| 4 | Java     | 49     |
| 5 | node.js  | 49     |
| 6 | Html     | 45     |
| 7 | python-3.x| 44 |
| 8 | javascript| 44  |
| 9 | Php      | 42     |
| 10 | Sql      | 41     |
| 11 | Jquery   | 38     |
| 12 | React.js | 36     |
| 13 | Android  | 35     |
| 14 | IOS      | 32     |
| 15 | Angular  | 32     |
| 16 | C++      | 31     |
| 17 | R        | 31     |
| 18 | Css      | 28     |
| 19 | Pandas   | 27     |
| 20 | Swift    | 26     |

5. PATH LENGTH

In a network graph the length of a path is the number of edges that the path contains and the average shortest path length is the sum of path lengths between all pairs of nodes normalized by n*(n-1) where n is the number of nodes in Graph, Figure 10 shows average path after grouping for tags. We can see that the AVG path length for all groups between (2-3 hops).
6. DEGREE DISTRIBUTION

Degree distribution is most important characteristic to understand the complex networks. Figure 11 shows the degree distribution of all graph and in Figure 12 we can see that the power low distribution of the degree sequence is generated for our graph groups. In addition, the graph shows that the network in scale free regime. The degrees sequence of undirected network is extracted from adjacency matrix of real network.

7. DIAMETER OF A NETWORK

Diameter it is the shortest distance between the two most distant nodes in the network. In another definition, Diameter is the longest path length between any pair of vertices, Figure 14 shows that the diameter for all groups is equal to 4.

8. NETWORK DENSITY

Network density describes the portion of the potential connections in a network that are actual connections. A “potential connection” is a connection that could potentially exist between two “nodes” – regardless of whether or not it actually does. Figure 15 shows the network density for groups of our network graph.

9. MODULARITY

Modularity is one measure of the structure of networks or graphs. It was designed to measure the strength of division of a network into modules (also called groups, clusters or communities), Figure 16 shows the modularity of our network graph groups.

Figure 11. Degree distribution for graph before grouping.

Figure 12. Degree distribution for graph after collected all groups.

Figure 13. Degree distribution for graph for each group.

Figure 14. Network Diameter for network graph after grouping.

Figure 15. Network Density for network graph after grouping.

Figure 16. Network Modularity for network graph after grouping.
10. BETWEENNESS CENTRALITY

Betweenness Centrality measures the extent to which a node is located in the shortest paths between other pairs of nodes. In Figure 17 we checked top 20 tags with the highest betweenness centrality.

| #  | Tag        | B.C |
|----|------------|-----|
| 1  | Mysql      | 0.1 |
| 2  | c#         | 0.08|
| 3  | java       | 0.08|
| 4  | python     | 0.08|
| 5  | sql        | 0.06|
| 6  | node.js    | 0.06|
| 7  | python-3.x  | 0.05|
| 8  | html       | 0.05|
| 9  | android    | 0.05|
| 10 | ios        | 0.04|
| 11 | javascript | 0.04|
| 12 | php        | 0.03|
| 13 | c++        | 0.03|
| 14 | swift      | 0.03|
| 15 | jquery     | 0.02|
| 16 | reactjs    | 0.02|
| 17 | pandas     | 0.02|
| 18 | r          | 0.02|
| 19 | angular    | 0.02|
| 20 | arrays     | 0.01|

Table 8. Top 20 tags with the highest betweenness centrality.

11. CLOSENESS CENTRALITY

The closeness centrality of a node is a measure of how ‘close’ a node is to all the other nodes in the network, we presented in Figure 18 top 20 tags with the highest closeness centrality.

| #  | Tag       | C.C |
|----|-----------|-----|
| 1  | mysql     | 0.58|
| 2  | python    | 0.56|
| 3  | arrays    | 0.56|
| 4  | json      | 0.56|
| 5  | html      | 0.55|
| 6  | javascript| 0.55|
| 7  | java      | 0.55|
| 8  | c#        | 0.55|
| 9  | java      | 0.55|
| 10 | node.js   | 0.55|
| 11 | android   | 0.54|
| 12 | regex     | 0.54|
| 13 | database  | 0.54|
| 14 | string    | 0.54|
| 15 | xml       | 0.54|
| 16 | php       | 0.53|
| 17 | api       | 0.53|
| 18 | loops     | 0.53|
| 19 | css       | 0.53|
| 20 | jquery    | 0.52|

Table 9. Top 20 tags with the highest closeness centrality.

12. EIGENVECTOR-BASED CENTRALITY

Eigenvector-based centrality measures express the importance of a node based on the importance of its neighbours. In Figure 19, we checked top 20 tags with the highest eigenvector centrality.
The resulting network edges are weighted; for example, if the tag java is found to co-exist with the tag android in 1000 questions, the weight of the edge between the nodes java and android in our graph will be 1000. In addition, we focused on this paper to study the connection between tags to identify clustered tags and isolated tags. We hope that we have presented a convincing case of the graph representation.

As future work, we plan to extend our experimentation to cover more real-world dataset tags and to experiment with larger number of dataset tags.

**REFERENCES**

E. Kassela, I. Konstantinou, and N. Koziiris, “Towards a Multi-engine Query Optimizer for Complex SQL Queries on Big Data,” Proc. - 2019 IEEE Int. Conf. Big Data, Big Data 2019, pp. 6095–6097, 2019.

O. Dawelbeit and R. McCrindle, “Efficient dictionary compression for processing RDF big data using google BigQuery,” 2016 IEEE Glob. Commun. Conf. GLOBECOM 2016 - Proc., 2016.

S. K. Routray, G. Sahin, J. R. F. Da Rocha, and A. N. Pinto, “Statistical analysis and modeling of shortest path lengths in optical transport networks,” J. Light. Technol., vol. 33, no. 13, pp. 2791–2801, 2015.

M. Newman. Networks: An Introduction. Oxford University Press, Inc., New York,NY, 2010.

Z. Zhang, W. Xiao, M. He, J. Xi, and Y. Mao, “Research and Analysis about the Length of Vertex-Degree Sequence of Complex Networks with Poisson Distribution,” Proc. - 2017 IEEE Int. Conf. Comput. Sci. Eng. IEEE/IFIP Int. Conf. Embed. Ubiquitous Comput. CSE EUC 2017, vol. 1, pp. 28–31, 2017.

Y. Sun, Y. Guan, Z. Wang, T. Zhang, C. Guo, and X. Li, “A diameter path based method for important node detection in complex network,” Proc. IECON 2017 - 43rd Annu. Conf. IEEE Ind. Electron. Soc., vol. 2017-Janua, pp. 5669–5674, 2017.

B. Martinez-Seis, X. Li, and X. Wang, “Measure community quality by attribute importance and density in social networks,” IEEE Int. Conf. Autom. Sci. Eng., vol. 2019-Augus, pp. 628–633, 2019.

A. Maulana, V. Gemmetto, D. Garlaschelli, I. Yevesyeva, and M. Emmerich, “Modularities maximization in multiplex network analysis using Many-Objective Optimization,” 2016 IEEE Symp. Ser. Comput. Intell. SSCI 2016, 2017.

M.O. Jackson. Social and Economic Networks. Princeton University Press, Princeton,NJ, 2008.

E.D. Kolaczyk and G. Csardi. Statistical Analysis of Network Data with R. Springer,2014.

D. Bates and M. Maechler. Matrix: Sparse and Dense Matrix Classes and Methods,2014. R package version 1.1-4.

L.C. Freeman. A set of measures of centrality based on betweenness. Sociometry,40(1):3[415].

### Table 10. Top 20 tags with the highest eigenvector centrality

| #  | Tag     | E.C  |
|----|---------|------|
| 1  | mysql   | 1    |
| 2  | node.js | 0.97 |
| 3  | python  | 0.94 |
| 4  | c#      | 0.93 |
| 5  | java    | 0.92 |
| 6  | javascript | 0.91 |
| 7  | html    | 0.89 |
| 8  | php     | 0.88 |
| 9  | jqueray | 0.8  |
| 10 | python-3.x | 0.79 |
| 11 | reactjs | 0.75 |
| 12 | sql     | 0.75 |
| 13 | arrays  | 0.67 |
| 14 | json    | 0.67 |
| 15 | angular | 0.67 |
| 16 | android | 0.66 |
| 17 | r       | 0.63 |
| 18 | ios     | 0.61 |
| 19 | css     | 0.6  |
| 20 | c++     | 0.59 |

### 13. CONCLUSION AND FUTURE WORK

In this paper, we discussed Network analysis for stack overflow tags and then we looked at some statistics after that; we used different centrality measures and concepts in network science for analysis process of stack overflow, according to the increasing presence of representations in modern data analysis. We carry out an analysis of the Stack Overflow tags viewed as a network or a graph.

We get some insight about the user communities by representing tags and their co-occurrences as a graph, where the graph nodes are the tags themselves, and an edge between two nodes exists if the corresponding tags are found together in the same Stack Overflow question.

The resulting network edges are weighted; for example, if the tag java is found to co-exist with the tag android in 1000