VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SOCIAL NETWORK ANALYSIS OF STACKOVERFLOW TAG DATASET

A Course Application Report

submitted as part of the course

Social Information Networks

CSE3021

School Of Computer Science and Engineering

VIT Chennai

FALL 2021-2022

Course Faculty : Dr. B Radhika Selvamani

Submitted By

Jill Jani (18BCE1302)

Parimala Chowdary (18BCE1309)

Abstract

Stack Overflow is a popular developer site where individuals ask and answer programming-related problems. Over 21 million questions have been asked, over 31 million answers have been supplied, and more than 80 million comments have been submitted.

We utilized a Stack Overflow Tag Network raw data set from the Kaggle website. This data collection comprises a network of technology tags from Developer Stories that will be used to identify popular and related technologies. We will also identify key network components that will assist us in retrieving language-specific information.

This network analysis confirms that technology trends are generally connected with one another. More specifically, it is proved that some technologies require some others in order to work effectively (e.g. jquery and html). Stackoverflow tags can be a useful tool in determining these trends. On the other hand, there are technologies that are completely independent, such as excel for instance. This possibly means that either they are about to be extinct or they should change somehow in order to be compatible with others.

The current network analysis has compatible results with the 2021 StackOverflow Trends Survey, highlighting that all widely used technologies have a lot in common as they may have a mutual starting node (e.g. Linux). This also shows how user response has a direct correlation with highly talked about technologies in stackoverflow.

This analysis provides insights and information required to understand, reach and attract developers. It also helps improve tech hiring, recruiting and developer marketing. Finally, it helps newcomers and developers track interest in programming languages and technologies based on the usage and popularity of tags.

Contents

1 Introduction

Each month, about 50 million people visit Stack Overflow to learn, share, and build their careers. It is estimated that 21 million of these people are pro- fessional developers and university-level students. The majority of our survey respondents are people who said they are professional developers or who code sometimes as part of their work, or are students preparing for such a career. About 4% of respondents code as a hobby but not as a profession, and just under 2% of respondents used to be professional developers but no longer are.

It is a vast community where millions of questions are asked and answered daily. In such applications, tagging is highly appreciated. Because they not only help us in easier understanding of the content available in the application but also, makes the discovery of new trends easy. These tags can also be effortlessly accessed by all the learners. Stackoverflow has a huge database of questions and answers. But it does not provide beginners with information about trending technologies and tech stacks followed by experts. Such situations can also make the newbies stressful as they are unsure of what is on high demand in the market workplace. But, this can be overcome by considering the graph' s nodes and edges. In this case the nodes are the tags that appear in a user's questions in stackoverflow.

2 Related Works

Stack Overflow is a vast community where millions of questions are asked and answered daily. In such applications, tagging is highly appreciated. Because they not only help us in easier understanding of the content available in the application but also, makes the discovery of new trends easily.

Using data from Stack Overflow, a study [1] examined the profile data of users in order to determine the relationship between a complete profile (one that has values for each profile element: website URL, location, about me, profile image and age) and their contribution to the network in terms of reputation scores and quality of question and answer posts. The analysis shows that most users do not have a complete profile, however the average reputation earned by users with complete profiles is significantly higher than that earned by users with incomplete profiles. In addition, users with complete profiles post higher quality questions and answers, hence are more useful to the network. It was also found that, of the five profile elements studied, location and about me have a higher correlation than the others. This research puts forward important profile elements in a typical QA social network.

Another study [3] with the objective of exploring the habits of users, more specif- ically, online was conducted. In doing so, it was attempted to present a new picture that focuses more on the human aspect of this online community. It neither focused on the technical aspects of the platform nor on the technical capabilities of developers, rather, the focus of this paper is to explore the hu- manistic point of view of technical users.

To determine how the communities have been evolving throughout years in their work, [5] analyzed all questions, answers, votes and tags from Stack Overflow between 2008 and 2020. We generated a series of user-technology interaction graphs and applied community detection algorithms to identify the biggest user communities for each year, to examine which technologies those communities incorporate, how they are interconnected and how they evolve through time. The biggest and most persistent communities were related to web development. In general, they found little movement between communities; users tend to either stay within the same community or not acquire any score at all. Community evolution reveals the popularity of different programming languages and frameworks on Stack Overflow over time. These findings give insight into the user community on Stack Overflow and reveal long-term trends on the software development industry.

While past research has considered the value of posts, often based on upvot- ing, little has examined the role of comments. Beyond value in explaining code, comments may offer new ways of looking at problems, clarifications of questions or answers, and socially supportive community interactions. To understand the role of comments, a content analysis was conducted to evaluate the key purposes of comments in another study [7]. A coding schema of nine comment categories was developed from open coding on a set of 40 posts and used to classify com- ments in a larger dataset of 2323 comments from 50 threads over a 6-month period. Results provide insight into the way the comments support learning, knowledge development, and the SO community, and the use and usefulness of the comment feature.

To understand the common challenges that users face when looking for in- formation about a specific domain a study was done to investigate common issues and/or misconceptions regarding security-related issues among developers [6]. A mixed-method analysis was conducted on the data obtained from Stack Overflow (SO), one of the most popular on-line QA sites for software devel- oper community to communicate, collaborate, and share information with one another. They adopted techniques from mining software repositories research paradigm and have employed topic modeling for analyzing security-related top- ics in SO dataset.

To show how academics/practitioners can get benefit from the valuable user- generated content shared on various online social networks, specifically from QA community SO for software development a study [2] conducted a compre- hensive literature review was conducted and 166 research papers on SO were categorized about software development from the inception of SO till June 2016. Most of the studies revolve around a limited number of software de- velopment tasks; approximately 70 percent of the papers used millions of posts data, applied basic machine learning methods, and conducted investigations semi-automatically and quantitative studies. Thus, future research should fo- cus on overcoming existing identified challenges and gaps.

2.1 History

It is quite interesting to conduct analysis on this data as it can give important insights about the popular and related technologies. This kind of analysis was seen in the following studies done by Edrees. It was an analysis for the stack overflow tags by using different criteria in network science, [4] one of the advantages of network analysis is that the complex of connections can be made clear. They started this work in the first step by extracting data from the dataset after that applied network concepts: node degree distribution, node importance (centrality measures).They also provided a brief demonstration of how graph networks and tools can be used to analyze semi-structured text as Tags.

3 Design

3.1 Data Overview

The dataset represents a network of technology tags from Developer Stories on the Stack Overflow online developer community website.
It consists of the following two files:

- stack network edges.csv contains links of the network, the source and target tech tags plus the value of the link between each pair. A graph consists of nodes and edges. In this case the nodes are the tags that appear in a developer’ s profile in stackoverflow ” Developer Stories” . If two tags appear on the same profile there’ s a tag between them.
  The dataset includes only a subset of tags used on Developer Stories, tags that were used by at least 0.5% of users and were correlated with another tag with a correlation coefficient above 0.1. This means that very sparsely used tags and tags that are not used with other tags were filtered out.

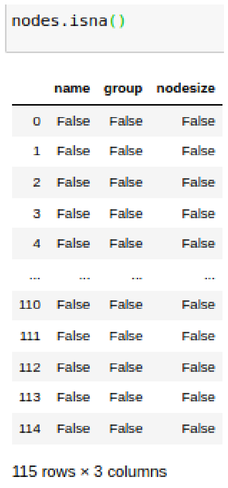
This file also includes the links of the network in 3 columns:

- Source tech tags
- Target tech tags
- Value of the the link between each pair, which means how correlated those two tags are (correlation coefficient * 100)

- stack network nodes.csv contains nodes of the network, the name of each node, which group that node belongs to (calculated via a cluster walktrap), and a node size based on how often that technology tag is used.
  This file also includes the links of the network in 3 columns:
  - name of tech tags
  - group in which each node belongs to (calculated via a cluster walk- trap)
  - nodesize which is proportional to how many developers have that tag in their developer story profile

3.2 Data Preprocessing

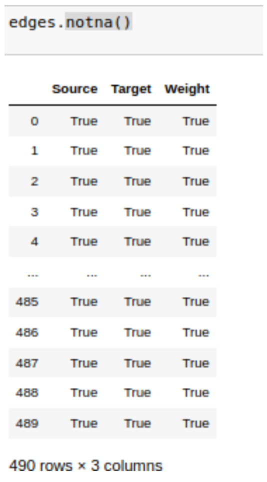
Removing null values



```
nodes.isna()
```

```
     name   group  nodesize
0    False  False  False
1    False  False  False
2    False  False  False
3    False  False  False
4    False  False  False
...  ...    ...    ...
110  False  False  False
111  False  False  False
112  False  False  False
113  False  False  False
114  False  False  False

115 rows × 3 columns
```

(a) nodes.csv



```
edges.notna()
```

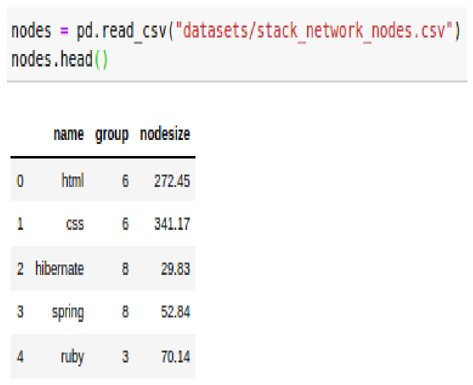
```
     Source  Target  Weight
0    True    True    True
1    True    True    True
2    True    True    True
3    True    True    True
4    True    True    True
...  ...     ...     ...
485  True    True    True
486  True    True    True
487  True    True    True
488  True    True    True
489  True    True    True

490 rows × 3 columns
```

(b) edges.csv

Figure 1: Null values in Dataset

3.2.1 Nodes Table

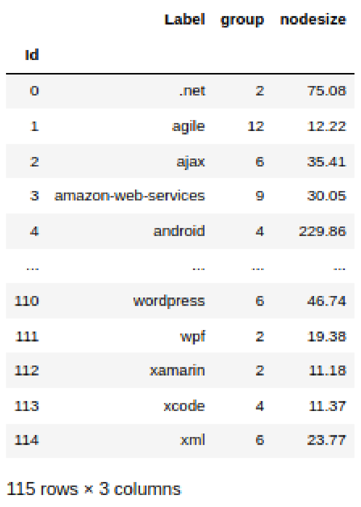
In the original dataset the Id of each tag was its own name. In order to be correctly represented as a graph, a unique Id was created for each tag.



```
nodes = pd.read_csv("datasets/stack_network_nodes.csv")
nodes.head()
```

	name	group	nodesize
0	html	6	272.45
1	css	6	341.17
2	hibernate	8	29.83
3	spring	8	52.84
4	ruby	3	70.14

(a) initial nodes table



	Label	group	nodesize
Id			
0	.net	2	75.08
1	agile	12	12.22
2	ajax	6	35.41
3	amazon-web-services	9	30.05
4	android	4	229.86
...	...	...	...
110	wordpress	6	46.74
111	wpf	2	19.38
112	xamarin	2	11.18
113	xcode	4	11.37
114	xml	6	23.77

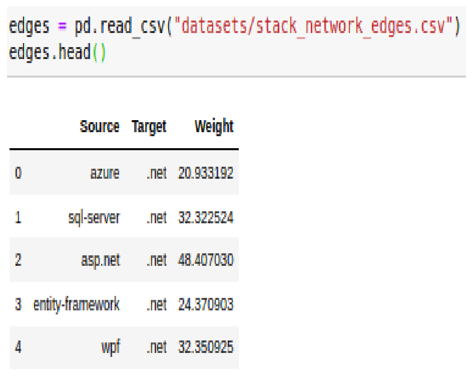
115 rows × 3 columns

(b) final nodes table

Figure 2: Preprocessed nodes.csv

3.2.2 Edges Table

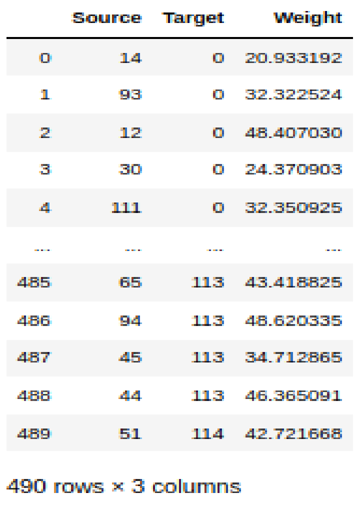
In this table, a new Id column was added, which was also the index of the table, and the column name was renamed as Label.



```
edges = pd.read_csv("datasets/stack_network_edges.csv")
edges.head()
```

	Source	Target	Weight
0	azure	.net	20.933192
1	sql-server	.net	32.322524
2	asp.net	.net	48.407030
3	entity-framework	.net	24.370903
4	wpf	.net	32.350925

(a) initial edges table



	Source	Target	Weight
0	14	0	20.933192
1	93	0	32.322524
2	12	0	48.407030
3	30	0	24.370903
4	111	0	32.350925
...	...	...	...
485	65	113	43.418825
486	94	113	48.620335
487	45	113	34.712865
488	44	113	46.365091
489	51	114	42.721668

490 rows × 3 columns

(b) final edges table

Figure 3: Preprocessed edges.csv

3.3 Network Analysis

3.3.1 Basic Topological Properties

The first representation of the graph is according to the dataset raw data. That means that each group has its own color and each node is sized according to the nodesize variable. The problem with this graph is that it is undirected and the Source & Target columns are not totally exploited.

- Number of nodes: 115
- Number of edges: 490
- Network diameter: 10 which is the biggest shortest path connecting two nodes.
- Average shortest path: 4.5 which means that two nodes have distance of 4.5 between them approximately.
- Network Density: 0.0374 , which means that is absolutely not a dense network for reasons being explicitly explained below.

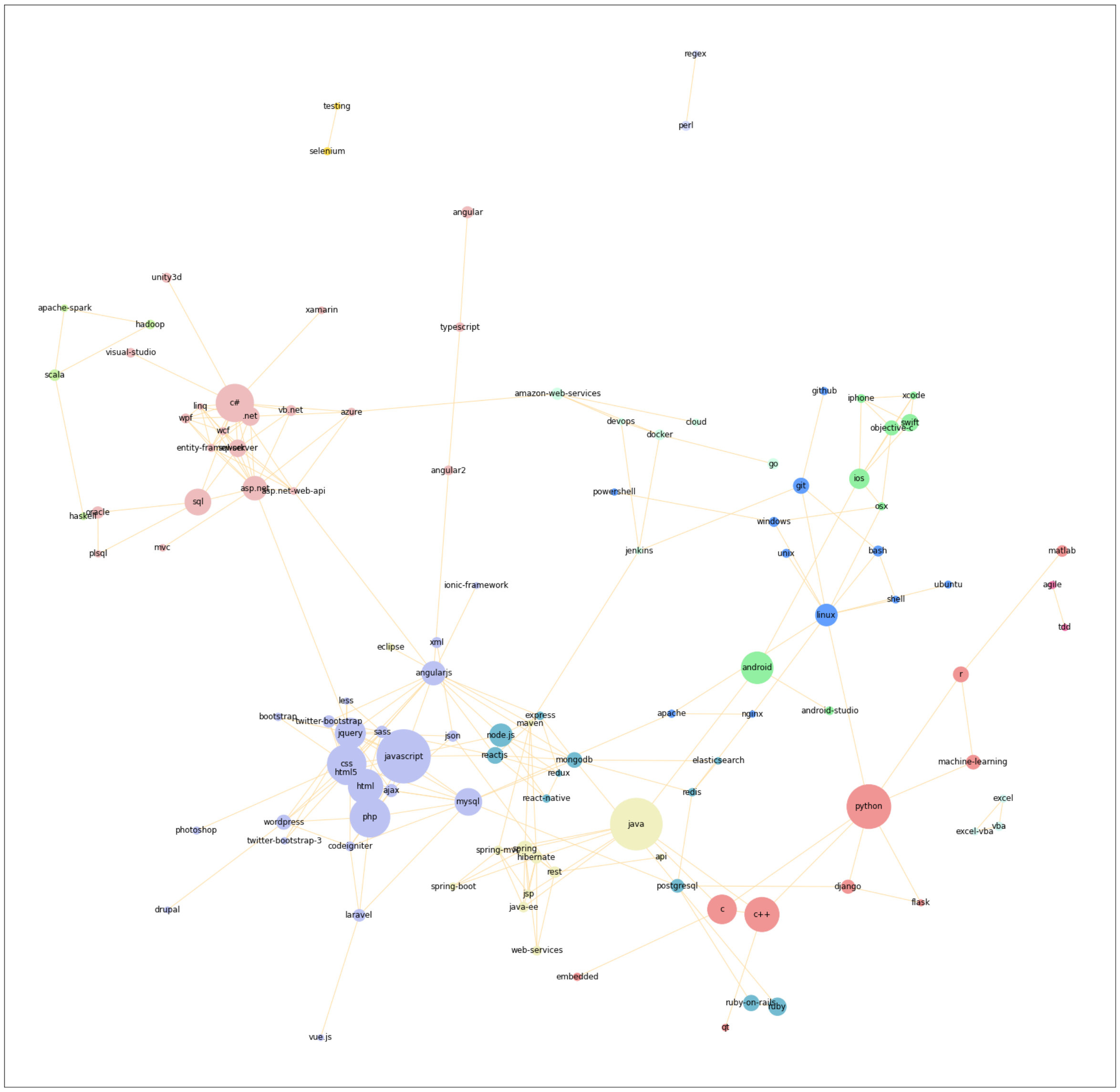


Figure 4: Network

Degree

The Average degree of the graph above is calculated to be 4.2609. To understand it better. We plotted a degree distribution graph

Density

On a scale of 0 to 1, not a very dense network, which comports with what is seen in the visualization. A 0 would mean that there are no connections at all, and a 1 would indicate that all possible edges are present (a perfectly connected network): this network is on the lower end of that scale, but still far from 0.

Shortest Path

It calculates the shortest possible series of nodes and edges that stand between any two nodes, something hard to see in large network visualizations. This measure is essentially finding friends-of-friends—if tag x is connected with y and w is connected with y, but not with x, then the shortest path between x and w is y. This is useful to determine how related two technologies are.

Since there is no shortest path between nodes of one component and nodes of another, nx.diameter() returns the "not connected" error. You can remedy this by first finding out if your Graph "is connected" (i.e. all one component) and, if not connected, finding the largest component and calculating diameter on that component alone.

The network diameter of this network' s largest component is 10: there is a path length of 10 between the two farthest-apart nodes in the network. Unlike density which is scaled from 0 to 1, it is difficult to know from this number alone whether 10 is a large or small diameter.

3.3.2 Centrality Measures

In network analysis, measures of the importance of nodes are referred to as centrality measures. Because there are many ways of approaching the question "Which nodes are the most important?" There are many different ways of calculating centrality.

Degree Centrality

The Degree Centrality shows how connected a node is. In this network the Degree Centrality shows how a tag (whether is a language or framework) is connected with all others tags and how all these technologies are finally connected.

Closeness Centrality

The Closeness Centrality shows how close a node is to the center of the network. This number is based on the length of the average shortest path between a tag and the rest tags of the network.

Betweenness Centrality

Betweenness Centrality shows how important a node is in terms of connecting other nodes. In this case which tags are part of the shortest paths between two others and eventually we could say which technologies may be prerequi- sites of others.

EigenVector Centrality

Eigenvector centrality shows how much a node is connected to other important nodes in the network. It is a weighted degree of centrality with a feedback boost for being connected with other important tags.

3.3.3 Clustering Effects

Clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties; this likelihood tends to be greater than the average probability of a tie randomly established between two nodes.

Two versions of this measure exist: the global and the local. The global version was designed to give an overall indication of the clustering in the network, whereas the local version gives an indication of the embeddedness of single nodes.

Transitivity

Triadic closure supposes that if two people know the same person, they are likely to know each other. If x is connected with both y and w, then y and w may very well know each other, completing a triangle in the visualization of three edges connecting x, y, and w. The number of these enclosed triangles in the network can be used to find clusters and communities of individuals that all know each other fairly well. One way to measure triadic closure is called clustering coefficient because of this clustering tendency, but it is also known as transitivity.Transitivity is the ratio of all triangles over all possible triangles. A possible triangle exists when one tag x knows two others (y and w). So transitivity, like density, expresses how interconnected a graph is in terms of a ratio of actual over possible connections.

3.3.4 Bridges, Communities and Cliques

Bridge

A bridge is a connection between two nodes that if it were to be deleted it would cause the total division of the main component into two separate components. This would help in identifying technologies connecting to separate domains.

Communities

Modularity is a measure of the structure of networks or graphs which measures the strength of division of a network into modules. Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules. Modularity community detection is used to detect technologies often used together.

Cliques

Cliques are useful in determining a group of individuals who interact with one another and share similar interests. Interacting with cliques is part of normative social development regardless of gender, ethnicity, or popularity.

3.4 Language Specific Analysis

For each programming language there's a tag in the network. E.g 'python' will refer to the python language. So we can check the cliques that contain that node. We can also visualize the subgraph containing that node and all its neighbors with a specified depth range.

For example, we can check the ego network for java with radius 2, which means that we get the subgraph containing java and all it's direct neighbors which are 1 edge away from java and also the nodes which are 2 steps away from java.

3.5 2021 Trends in Stackoverflow

Using data from Stackoverflow insights survey report we try to look for new trends in 2021. This report is based on a survey of 83,439 software developers from 181 countries around the world. This is the number of responses considered "qualified" for analytical purposes based on time spent on the full, completed survey. Many questions were only shown to respondents based on their previous answers. For example, questions about jobs and work were only shown to those who said they were working in a job.

4 Results & Discussion

4.1 Network Analysis

4.1.1 Basic Topological Properties

Degree

The average degree of this network is 4.2609. The tag with the highest degree is jquery with 16. The degree distribution shows us that most of the tags have degree 1 and 3.

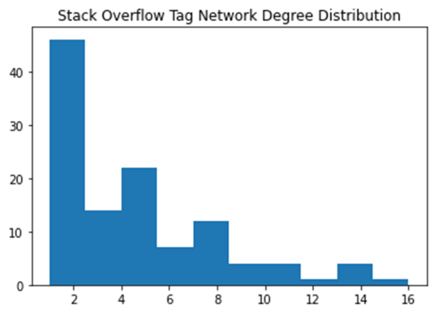
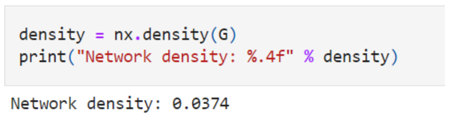


Figure 5: Degree Distribution graph

The average weighted degree is 148.995, which is higher than average degree, because the weight of each link is considered. It is natural for jQuery to be the most connected node due to the fact that it is a library for JavaScript, which is a broadly used and on high demand web development language.

Density

On a scale of 0 to 1, not a very dense network, which comports with what is seen in the visualization. A 0 would mean that there are no connections at all, and a 1 would indicate that all possible edges are present (a perfectly connected network): this network is on the lower end of that scale, but still far from 0.



```
density = nx.density(G)
print("Network density: %.4f" % density)
```

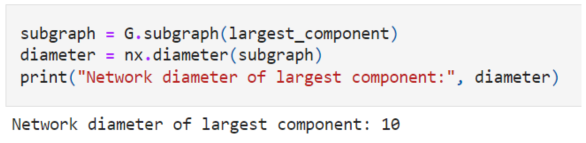
```
Network density: 0.0374
```

Figure 6: Network Density

Shortest Path

It calculates the shortest possible series of nodes and edges that stand between any two nodes, something hard to see in large network visualizations. This measure is essentially finding friends-of-friends—if tag x is connected with y and w is connected with y, but not with x, then the shortest path between x and w is y. This is useful to determine how related two technologies are.

Since there is no shortest path between nodes of one component and nodes of another, nx.diameter() returns the "not connected" error. You can remedy this by first finding out if your Graph "is connected" (i.e. all one component) and, if not connected, finding the largest component and calculating diameter on that component alone.



```
subgraph = G.subgraph(largest_component)
diameter = nx.diameter(subgraph)
print("Network diameter of largest component:", diameter)
```

```
Network diameter of largest component: 10
```

Figure 7: Network Shortest Path

The network diameter of this network’ s largest component is 10: there is a path length of 10 between the two farthest-apart nodes in the network. Unlike density which is scaled from 0 to 1, it is difficult to know from this number alone whether 10 is a large or small diameter.

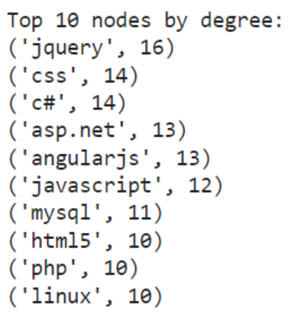
4.1.2 Centrality Measures

In network analysis, measures of the importance of nodes are referred to as centrality measures. Because there are many ways of approaching the question “Which nodes are the most important?” There are many different ways of calculating centrality.

Degree Centrality

The Degree Centrality shows how connected a node is. In this network the Degree Centrality shows how a tag (whether is a language or framework) is connected with all others tags and how all these technologies are finally connected.

Top 10 tags with the highest degree are the followings:



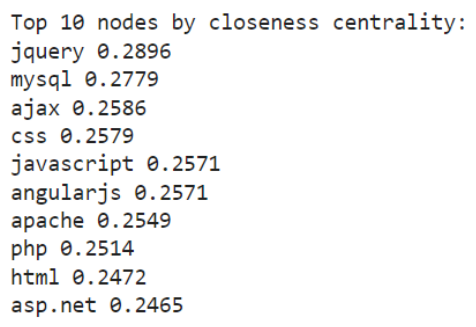
```
Top 10 nodes by degree:
('jquery', 16)
('css', 14)
('c#', 14)
('asp.net', 13)
('angularjs', 13)
('javascript', 12)
('mysql', 11)
('html5', 10)
('php', 10)
('linux', 10)
```

Figure 8: Top 10 tags with the highest degree centrality

Closeness Centrality

The Closeness Centrality shows how close a node is to the center of the network. This number is based on the length of the average shortest path between a tag and the rest tags of the network.

Top 10 tags with the highest closeness centrality degree are the followings:



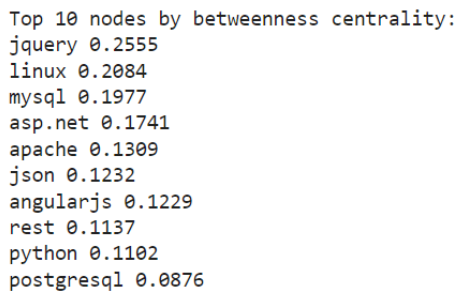
```
Top 10 nodes by closeness centrality:
jquery 0.2896
mysql 0.2779
ajax 0.2586
css 0.2579
javascript 0.2571
angularjs 0.2571
apache 0.2549
php 0.2514
html 0.2472
asp.net 0.2465
```

Figure 9: Top 10 tags with the highest closeness centrality

Top 10 tags with the highest closeness centralities are all connected with the most central node, which is jQuery, as we already mentioned. All are technologies (programming languages, frameworks etc.) which are working complementary(html,css) or using jQuery to work (JavaScript).

Betweenness Centrality

Betweenness Centrality shows how important a node is in terms of connecting other nodes. In this case which tags are part of the shortest paths between two others and eventually we could say which technologies may be prerequisites of others. Top 10 tags with the highest betweenness centrality degree are the followings:



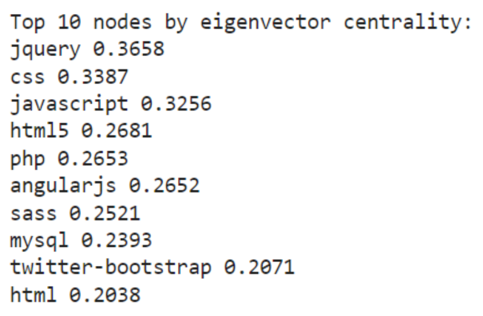
```
Top 10 nodes by betweenness centrality:
jquery 0.2555
linux 0.2084
mysql 0.1977
asp.net 0.1741
apache 0.1309
json 0.1232
angularjs 0.1229
rest 0.1137
python 0.1102
postgresql 0.0876
```

Figure 10: Top 10 tags with the highest betweenness centrality

Linux has the second highest betweenness centrality degree and that is because of being one of the most broadly used Operational Systems, especially in the developers community. There are daily developed and come in production new applets, plug-ins and other technologies compatible for Linux OS.

EigenVector Centrality

Eigenvector centrality shows how much a node is connected to other important nodes in the network. It is a weighted degree of centrality with a feedback boost for being connected with other important tags. Top 10 tags with the highest eigenvector centrality degree are the followings:



```
Top 10 nodes by eigenvector centrality:
jquery 0.3658
css 0.3387
javascript 0.3256
html5 0.2681
php 0.2653
angularjs 0.2652
sass 0.2521
mysql 0.2393
twitter-bootstrap 0.2071
html 0.2038
```

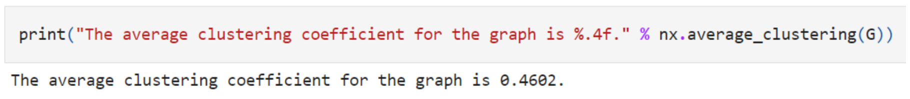
Figure 11: Top 10 tags with the highest eigenvector centrality

Technologies above concerns all those frameworks and languages which are using jQuery as a prerequisite to work correctly on the platform implemented. For example, twitter-bootstrap is the most popular HTML, CSS, and JS library in the world and requires jQuery.

4.1.3 Clustering Effects

Clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties; this likelihood tends to be greater than the average probability of a tie randomly established between two nodes.

The average clustering coefficient of this network is 0.4602, which means that there is 46.02% possibility that 2 tags will be connected if they are connected to a tag.



```
print("The average clustering coefficient for the graph is %.4f." % nx.average_clustering(G))
```

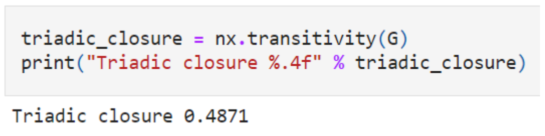
```
The average clustering coefficient for the graph is 0.4602.
```

Figure 12: Average clustering coefficient

Two versions of this measure exist: the global and the local. The global version was designed to give an overall indication of the clustering in the network, whereas the local version gives an indication of the embeddedness of single nodes.

Transitivity

Triadic closure supposes that if two people know the same person, they are likely to know each other. If x is connected with both y and w, then y and w may very well know each other, completing a triangle in the visualization of three edges connecting x, y, and w.



```
triadic_closure = nx.transitivity(G)
print("Triadic closure %.4f" % triadic_closure)
```

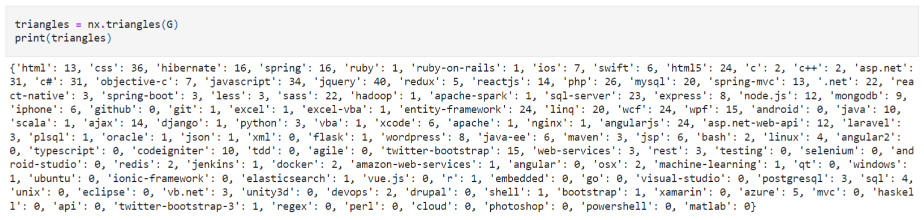
```
Triadic closure 0.4871
```

Figure 13: Transitivity (Triadic Closure)

The number of these enclosed triangles in the network can be used to find clusters and communities of individuals that all know each other fairly well. One way to measure triadic closure is called clustering coefficient because of this clustering tendency, but it is also known as transitivity.

The total triangles of this network is 239.

Transitivity is the ratio of all triangles over all possible triangles. A possible triangle exists when one tag x knows two others (y and w). So transitivity, like density, expresses how interconnected a graph is in terms of a ratio of actual over possible connections.



```
triangles = nx.triangles(G)
print(triangles)
```

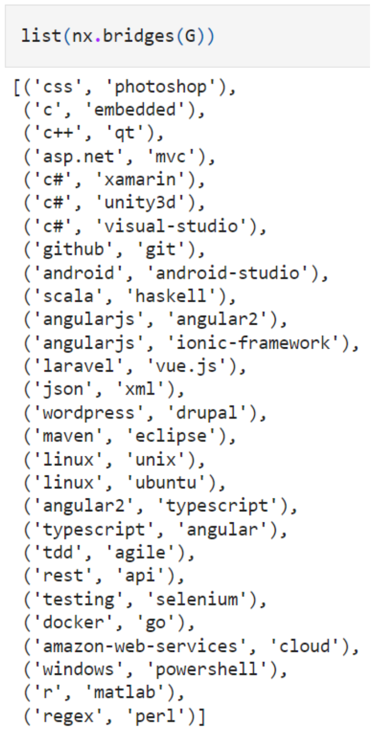
```
{'html': 13, 'css': 36, 'hibernate': 16, 'spring': 16, 'ruby': 1, 'ruby-on-rails': 1, 'ios': 7, 'swift': 6, 'html5': 24, 'c': 2, 'c++': 2, 'asp.net':
31, 'c#': 31, 'objective-c': 7, 'javascript': 34, 'jquery': 40, 'redux': 5, 'reactjs': 14, 'php': 26, 'mysql': 20, 'spring-mvc': 13, '.net': 22, 'rea
ct-native': 3, 'spring-boot': 3, 'less': 3, 'sass': 22, 'hadoop': 1, 'apache-spark': 1, 'sql-server': 23, 'express': 8, 'node.js': 12, 'mongodb': 9,
'iphone': 6, 'github': 0, 'git': 1, 'excel': 1, 'excel-vba': 1, 'entity-framework': 24, 'linq': 20, 'wcf': 24, 'wpf': 15, 'android': 0, 'java': 10,
'scala': 1, 'ajax': 14, 'django': 1, 'python': 3, 'vba': 1, 'xcode': 6, 'apache': 1, 'nginx': 1, 'angularjs': 24, 'asp.net-web-api': 12, 'laravel':
3, 'plsql': 1, 'oracle': 1, 'json': 1, 'xml': 0, 'flask': 1, 'wordpress': 8, 'java-ee': 6, 'maven': 3, 'jsp': 6, 'bash': 2, 'linux': 4, 'angular2':
0, 'typescript': 0, 'codeigniter': 10, 'tdd': 0, 'agile': 0, 'twitter-bootstrap': 15, 'web-services': 3, 'rest': 3, 'testing': 0, 'selenium': 0, 'and
roid-studio': 0, 'redis': 2, 'jenkins': 1, 'docker': 2, 'amazon-web-services': 1, 'angular': 0, 'osx': 2, 'machine-learning': 1, 'qt': 0, 'windows':
1, 'ubuntu': 0, 'ionic-framework': 0, 'elasticsearch': 1, 'vue.js': 0, 'r': 1, 'embedded': 0, 'go': 0, 'visual-studio': 0, 'postgresql': 3, 'sql': 4,
'unix': 0, 'eclipse': 0, 'vb.net': 3, 'unity3d': 0, 'devops': 2, 'drupal': 0, 'shell': 1, 'bootstrap': 1, 'xamarin': 0, 'azure': 5, 'mvc': 0, 'haskel
l': 0, 'api': 0, 'twitter-bootstrap-3': 1, 'regex': 0, 'perl': 0, 'cloud': 0, 'photoshop': 0, 'powershell': 0, 'matlab': 0}
```

Figure 14: All triangles in network

4.1.4 Bridges, Communities and Cliques

Bridge

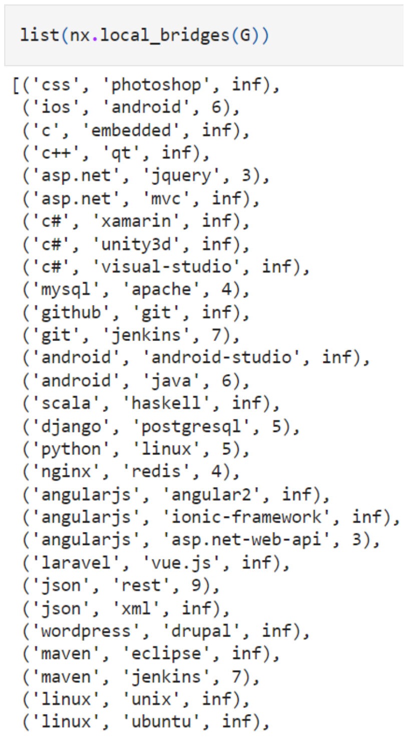
A bridge is a connection between two nodes that if it were to be deleted it would cause the total division of the main component into two separate components. This would help in identifying technologies connecting to separate domains.



```
list(nx.bridges(G))
```

```
[('css', 'photoshop'),
 ('c', 'embedded'),
 ('c++', 'qt'),
 ('asp.net', 'mvc'),
 ('c#', 'xamarin'),
 ('c#', 'unity3d'),
 ('c#', 'visual-studio'),
 ('github', 'git'),
 ('android', 'android-studio'),
 ('scala', 'haskell'),
 ('angularjs', 'angular2'),
 ('angularjs', 'ionic-framework'),
 ('laravel', 'vue.js'),
 ('json', 'xml'),
 ('wordpress', 'drupal'),
 ('maven', 'eclipse'),
 ('linux', 'unix'),
 ('linux', 'ubuntu'),
 ('angular2', 'typescript'),
 ('typescript', 'angular'),
 ('tdd', 'agile'),
 ('rest', 'api'),
 ('testing', 'selenium'),
 ('docker', 'go'),
 ('amazon-web-services', 'cloud'),
 ('windows', 'powershell'),
 ('r', 'matlab'),
 ('regex', 'perl')]
```

(b) bridges



```
list(nx.local_bridges(G))
```

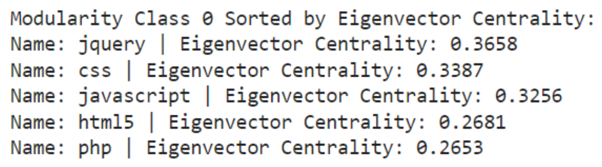
```
[('css', 'photoshop', inf),
 ('ios', 'android', 6),
 ('c', 'embedded', inf),
 ('c++', 'qt', inf),
 ('asp.net', 'jquery', 3),
 ('asp.net', 'mvc', inf),
 ('c#', 'xamarin', inf),
 ('c#', 'unity3d', inf),
 ('c#', 'visual-studio', inf),
 ('mysql', 'apache', 4),
 ('github', 'git', inf),
 ('git', 'jenkins', 7),
 ('android', 'android-studio', inf),
 ('android', 'java', 6),
 ('scala', 'haskell', inf),
 ('django', 'postgresql', 5),
 ('python', 'linux', 5),
 ('nginx', 'redis', 4),
 ('angularjs', 'angular2', inf),
 ('angularjs', 'ionic-framework', inf),
 ('angularjs', 'asp.net-web-api', 3),
 ('laravel', 'vue.js', inf),
 ('json', 'rest', 9),
 ('json', 'xml', inf),
 ('wordpress', 'drupal', inf),
 ('maven', 'eclipse', inf),
 ('maven', 'jenkins', 7),
 ('linux', 'unix', inf),
 ('linux', 'ubuntu', inf),
```

(b) local bridges

Figure 15: Bridges in network

Communities

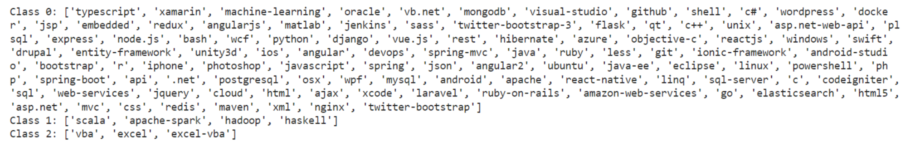
Modularity is a measure of the structure of networks or graphs which measures the strength of division of a network into modules.



```
Modularity Class 0 Sorted by Eigenvector Centrality:
Name: jquery | Eigenvector Centrality: 0.3658
Name: css | Eigenvector Centrality: 0.3387
Name: javascript | Eigenvector Centrality: 0.3256
Name: html5 | Eigenvector Centrality: 0.2681
Name: php | Eigenvector Centrality: 0.2653
```

Figure 16: Nodes with highest modularity in Class 0

Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules. Modularity community detection is used to detect technologies often used together.



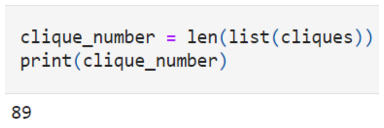
```
Class 0: ['typescript', 'xamarin', 'machine-learning', 'oracle', 'vb.net', 'mongodb', 'visual-studio', 'github', 'shell', 'c#', 'wordpress', 'docke
r', 'jsp', 'embedded', 'redux', 'angularjs', 'matlab', 'jenkins', 'sass', 'twitter-bootstrap-3', 'flask', 'qt', 'c++', 'unix', 'asp.net-web-api', 'pl
sql', 'express', 'node.js', 'bash', 'wcf', 'python', 'django', 'vue.js', 'rest', 'hibernate', 'azure', 'objective-c', 'reactjs', 'windows', 'swift',
'drupal', 'entity-framework', 'unity3d', 'ios', 'angular', 'devops', 'spring-mvc', 'java', 'ruby', 'less', 'git', 'ionic-framework', 'android-studi
o', 'bootstrap', 'r', 'iphone', 'photoshop', 'javascript', 'spring', 'json', 'angular2', 'ubuntu', 'java-ee', 'eclipse', 'linux', 'powershell', 'ph
p', 'spring-boot', 'api', '.net', 'postgresql', 'osx', 'wpf', 'mysql', 'android', 'apache', 'react-native', 'linq', 'sql-server', 'c', 'codeigniter',
'sql', 'web-services', 'jquery', 'cloud', 'html', 'ajax', 'xcode', 'laravel', 'ruby-on-rails', 'amazon-web-services', 'go', 'elasticsearch', 'html5',
'asp.net', 'mvc', 'css', 'redis', 'maven', 'xml', 'nginx', 'twitter-bootstrap']
Class 1: ['scala', 'apache-spark', 'hadoop', 'haskell']
Class 2: ['vba', 'excel', 'excel-vba']
```

Figure 17: All communitites formed using Modularity Community detection

Cliques

In general we consider cliques as groups of people who are closely connected to each other but not connected to people outside the group. In network theory a clique is defined as a maximal complete subgraph of a graph where each node is connected to all the other nodes. The word ' maximal' means that if we add another node to the clique the clique will cease to be a clique. nx.find cliques finds all the cliques in a network. We can also extract all the cliques from the tag network.

The Stack Overflow tag network has 89 cliques according to networkx.



```
clique_number = len(list(cliques))
print(clique_number)
```

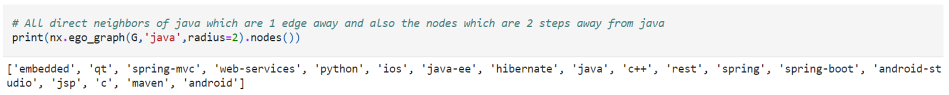
```
89
```

Figure 18: NUmber of Cliques in network

4.2 Language Specific Subgraphs

For each programming language there's a tag in the network. E.g 'python' will refer to the python language. So we can check the cliques that contain that node. We can also visualize the subgraph containing that node and all its neighbors with a specified depth range.

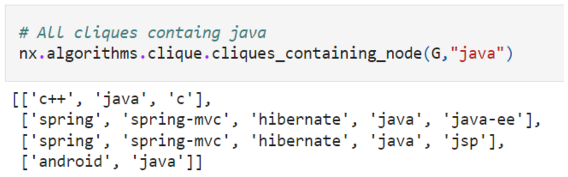
For example, we can check the ego network for java with radius 2, which means that we get the subgraph containing java and all it's direct neighbors which are 1 edge away from java and also the nodes which are 2 steps away from java.



```
# All direct neighbors of java which are 1 edge away and also the nodes which are 2 steps away from java
print(nx.ego_graph(G,'java',radius=2).nodes())
```

```
['embedded', 'qt', 'spring-mvc', 'web-services', 'python', 'ios', 'java-ee', 'hibernate', 'java', 'c++', 'rest', 'spring', 'spring-boot', 'android-st
udio', 'jsp', 'c', 'maven', 'android']
```

Figure 19: Nodes in Ego network of java with radius 2

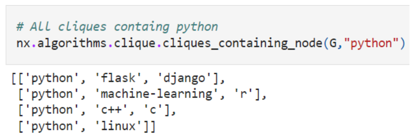


```
# All cliques containg java
nx.algorithms.clique.cliques_containing_node(G,"java")
```

```
[['c++', 'java', 'c'],
 ['spring', 'spring-mvc', 'hibernate', 'java', 'java-ee'],
 ['spring', 'spring-mvc', 'hibernate', 'java', 'jsp'],
 ['android', 'java']]
```

Figure 20: Cliques containing node java

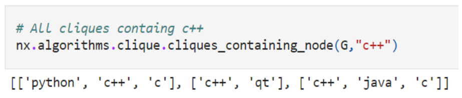
These subgraphs are called Ego networks and can be used for checking shortest paths or generally conducting analysis of who is connected to whom, but cliques are helpful because it shows us the data in a more granular way.



```
# All cliques containg python
nx.algorithms.clique.cliques_containing_node(G,"python")
```

```
[['python', 'flask', 'django'],
 ['python', 'machine-learning', 'r'],
 ['python', 'c++', 'c'],
 ['python', 'linux']]
```

(c) initial nodes table



```
# All cliques containg c++
nx.algorithms.clique.cliques_containing_node(G,"c++")
```

```
[['python', 'c++', 'c'], ['c++', 'qt'], ['c++', 'java', 'c']]
```

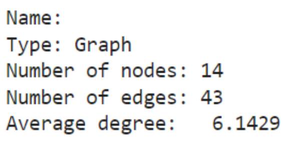
(b) final nodes table

Figure 21:Cliques containing nodes python and C++

By using nodes contained in three largest cliques of the network dataset we conclude that the tags being includes in max cliques of this network are the followings:

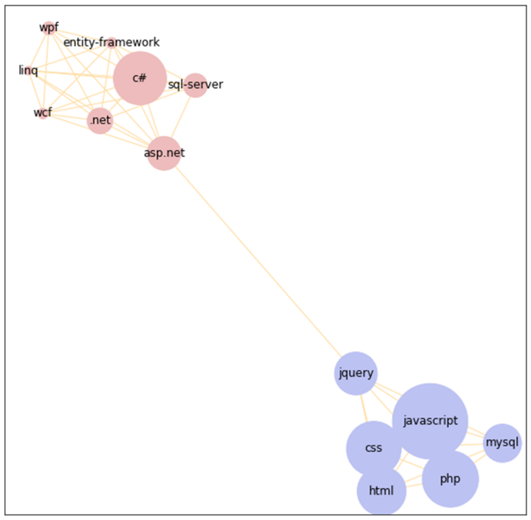
1. .net
2. ajax
3. asp.net
4. c#
5. css
6. entity-framework
7. javascript
8. jquery
9. linq
10. mysql
11. php
12. sql-server
13. wcf
14. wpf

We create a subgraph with 14 nodes, 43 edges and an average degree of 6.1429.



```
Name:
Type: Graph
Number of nodes: 14
Number of edges: 43
Average degree:   6.1429
```

(a) initial nodes table



(b) final nodes table

Figure 22: Max cliques Subgraph

4.3 2021 Trends in Stackoverflow

Using data from Stackoverflow insights survey report we try to look for new trends in 2021. This report is based on a survey of 83,439 software developers from 181 countries around the world. This is the number of responses considered "qualified" for analytical purposes based on time spent on the full, completed survey. Many questions were only shown to respondents based on their previous answers. For example, questions about jobs and work were only shown to those who said they were working in a job.

Most Popular Technologies

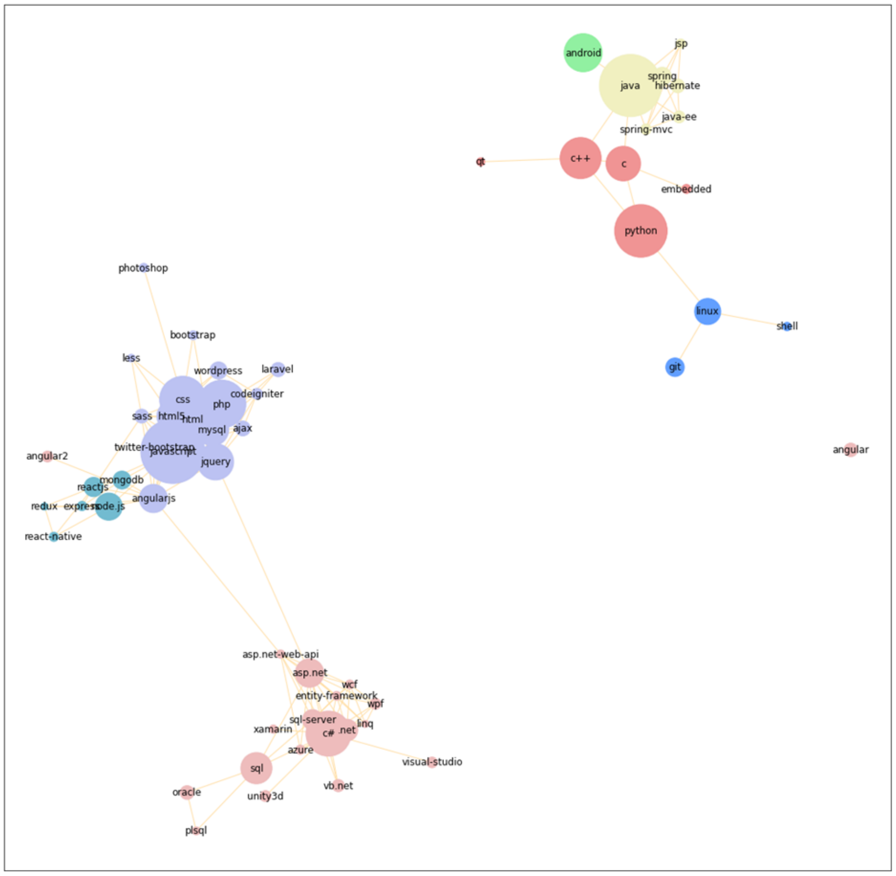


Figure 23: Most Popular Technologies Network

We noticed that, for instance, if one wants to become acquainted with JavaScript, he definitely should be first familiar with all of the javascript's tag neighbors (css, htl5, jquery etc.)

Most loved and wanted technologies

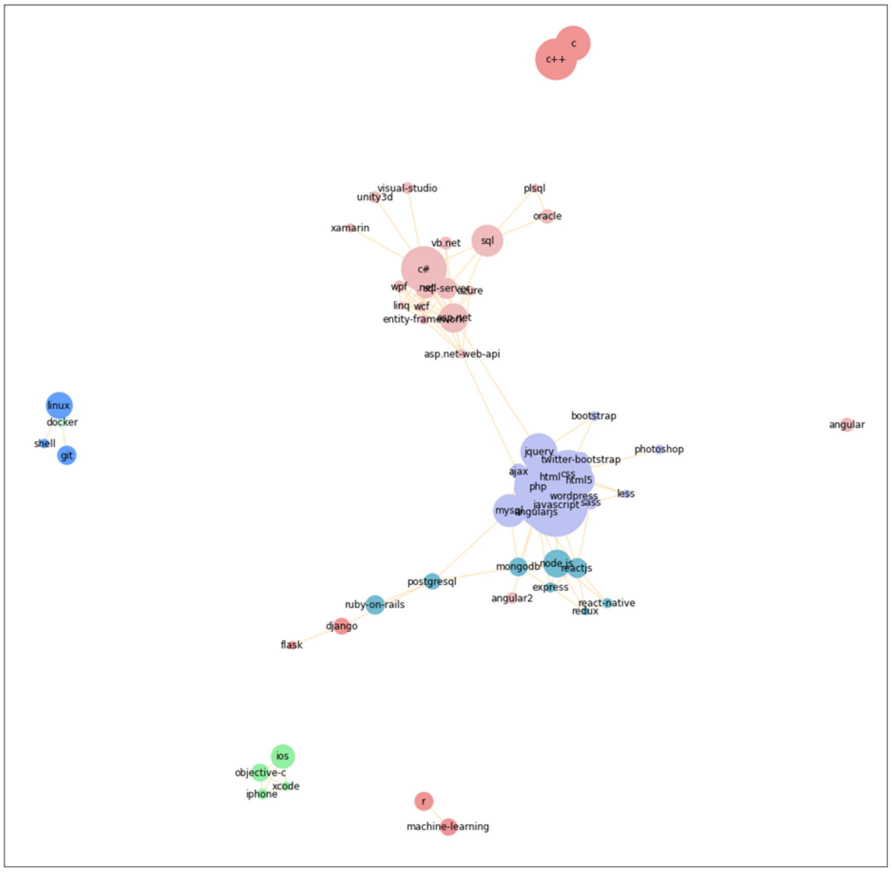


Figure 24: Most Loved and wanted Technologies Network

It can be seen from the above graph that c++, c# and javascript etc. are amongst the most loved technologies. Also, it can be observed that Angular - which will lose its Long Term Support by 31 December 2021; is detached from javascript. The graph also shows a set of technologies that you would take up when picking a domain. Anyone trying to learn docker for example, will need to have some knowledge about linux, shell and git as well.

Top paying technologies of 2021

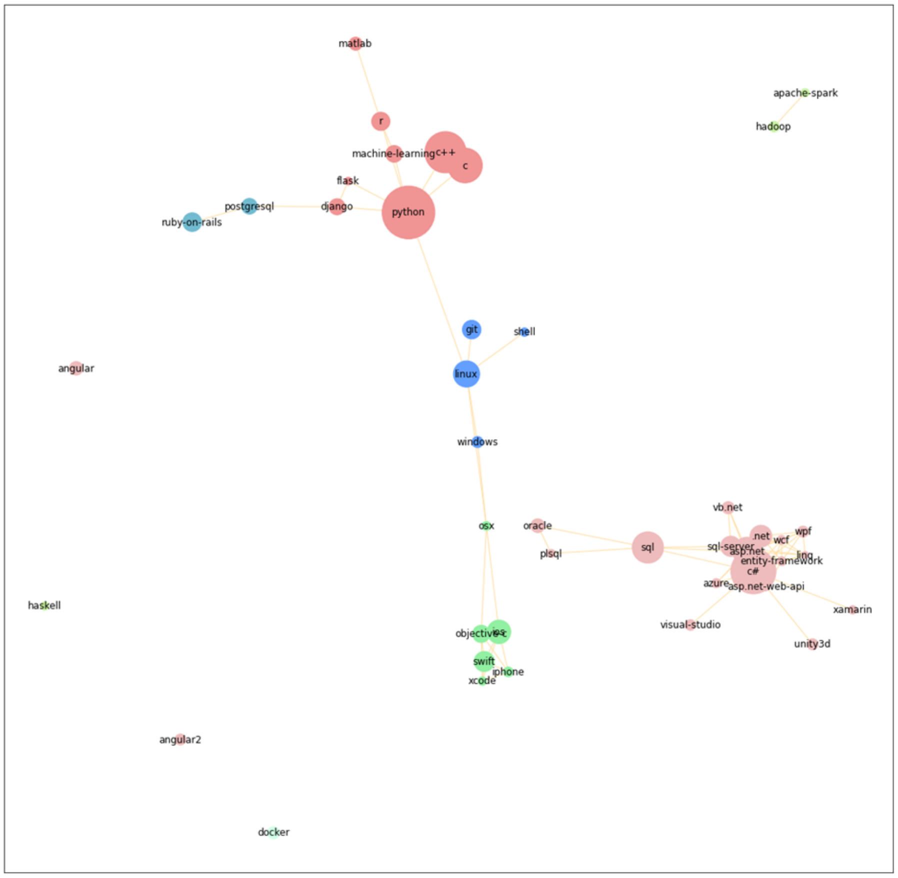


Figure 25: Top paying Technologies Network

It can be seen that newer technologies like Golang, Ruby, Typescript have quickly become a top paying technology to learn. Again, the communities can be used to understand commonly worked with technologies for a given technology.

5 Conclusion & Future Work

This network analysis confirms that technology trends are generally connected with one another. More specifically, it is proved that some technologies require some others in order to work effectively (e.g. jquery and html). Stackoverflow tags can be a useful tool in determining these trends. On the other hand, there are technologies that are completely independent, such as excel for instance. This possibly means that either they are about to be extinct or they should change somehow in order to be compatible with others.

Last but not least, the current network analysis has compatible results with the 2021 StackOverflow Trends Survey, highlighting that all widely used technologies have a lot in common as they may have a mutual starting node (e.g. Linux). This also shows how user response has a direct correlation with highly talked about technologies in stackoverflow.

This analysis provides insights and information required to understand, reach and attract developers. It also helps improve tech hiring, recruiting and developer marketing. Finally, it helps newcomers and developers track interest in programming languages and technologies based on the usage and popularity of tags.

Acknowledgments

We wish to express our sincere thanks and deep sense of gratitude to Dr. Nithyanandam P, Head of the Department (HoD), B.Tech Computer Science and Engineering .SCSE, VIT Chennai for his consistent encouragement and valuable guidance offered throughout the courses.

We are extremely grateful to Dr. Ganesan R, Dean of the School of Computer Science & Engineering, VIT Chennai for extending the facilities of the school to provide knowledge and for his unstinting support.

We express sincere thanks to Dr. Geetha S, Associate Dean of the School of Computer Science & Engineering, VIT Chennai for her support throughout the course of our degree.

We would like to express our gratitude to Dr. Parvathi R and Dr. Radhika Selvamani,School of Computer Science & Engineering, VIT Chennai for their support throughout the course of our project.

We thank our parents, family, and friends for being extremely supportive and encouraging throughout the course of the Project.

References

[1] Ifeoma Adaji and Julita Vassileva. Towards understanding user participation in stack overflow using profile data. pages 3 – 13, 10 2016.

[2] Arshad Ahmad, Chong Feng, Shi Ge, and Abdallah Yousif. A survey on mining stack overflow: question and answering (qa) community. *Data Tech- nologies and Applications*, 52, 02 2018.

[3] Tanveer Ahmed and Abhishek Srivastava. Understanding and evaluating the behavior of technical users. a study of developer interaction at stackoverflow. *Human-centric Computing and Information Sciences*, 7, 12 2017.

[4] Zahir Edrees. Network analysis of the stack overflow tags. volume XLIV- 4/W3-2020, 12 2020.

[5] Iraklis Moutidis and Hywel T. P. Williams. Community evolution on stack overflow. *PLOS ONE*, 16:1 – 23, 06 2021.

[6] Muhammad Rahman. *An Empirical Case Study on Stack Overflow to Ex- plore Developers' Security Challenges*. PhD thesis, 11 2016.

[7] Subhasree Sengupta and Caroline Haythornthwaite. Learning with com- ments: An analysis of comments and community on stack overflow. 01 2020.

APPENDIX

Code for this Project is available on github: https://github.com/jill-jani/Stack-Overflow-Network-Analysis