ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

Stack Overflow Tag Network

stackoverflow

Network Analysis
Supervisor Professor: Dimitrios Pournarakis

Rafaila Galanopoulou 8160018

Contents

1 Idea

Searching for a network dataset that would be interesting to analyse, I found this network (edges and nodes) of Stack Overflow tags based on Developer Stories. "Stack Overflow is a question and answer site for professional and enthusiast programmers. It is a privately held website, the flagship site of the Stack Exchange Network, created in 2008 by Jeff Atwood and Joel Spolsky. It features questions and answers on a wide range of topics in computer programming" Wikipedia.
According to its creators the data team at Stack Overflow, they spend a lot of their time and energy thinking about tech ecosystems and how technologies are related to each other. They discover that a way to get at this idea of relationships between technologies is tag correlations, how often technology tags at Stack Overflow appear together relative to how often they appear separately. One place we see developers using tags at Stack Overflow is on their Developer Stories, or professional profiles/CVs/resumes. If we are interested in how technologies are connected and how they are used together, developers' own descriptions of their work and careers is a great place to get that.
Each month, about 50 million people visit Stack Overflow to learn, share, and build their careers. We estimate that 21 million of these people are professional developers and university-level students. The majority of our survey respondents this year were 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. According to the Stack Overflow Survey.

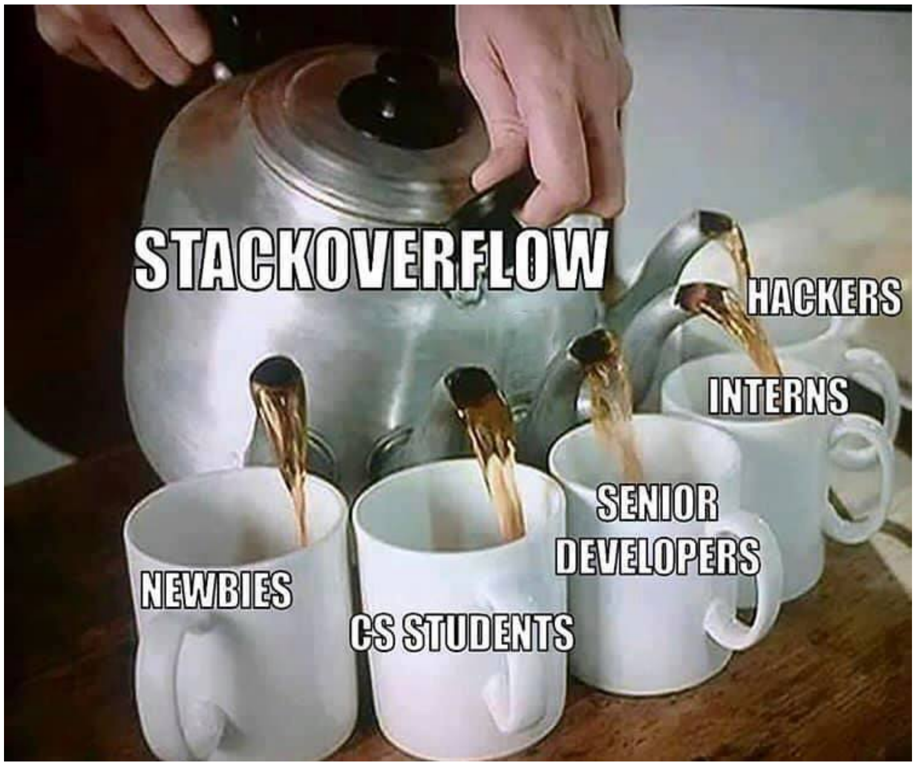


Figure 1: The role of Stack Overflow in our lives
Source

The main reason that I selected this worldwide known and used platform network stems from my recent search for an internship placement. It was very stressful to search 24/7 what is on high demand in the market workplace and if my studies are enough in order to start a career. As a junior level developer, I found quite interesting to see all the relationships between in-trend technologies but most importantly to answer the question "Which are the prerequisite technological background of learning each technology?".
The Python notebooks created for Data Preprocessing and Network Analysis are available here.

2 Dataset

2.1 Source

The dataset selected for the Network Analysis is the Stack Overflow Tag Network and can be found in Kaggle.

2.2 Data Overview

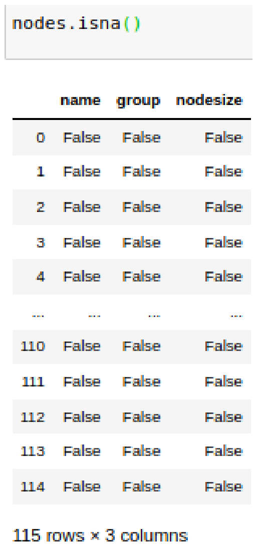
The dataset represents a network of technology tags from Developer Stories on the Stack Overflow online developer community website.
It is consisted the following two files:

- stack_network_edges.csv contains links of the network, the source and target tech tags plus the value of the 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 include 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

2.3 Data Preprocessing

The dataset was already cleaned so no null or missing values exist and, thus, no cleaning was necessary. Although using Python, I checked it, as it seems below.

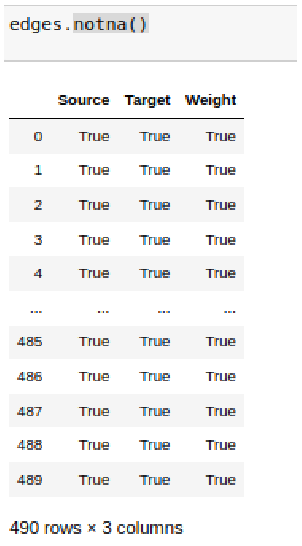


```
nodes.isna()
```

	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
It would be True if there were any NaN values



```
edges.notna()
```

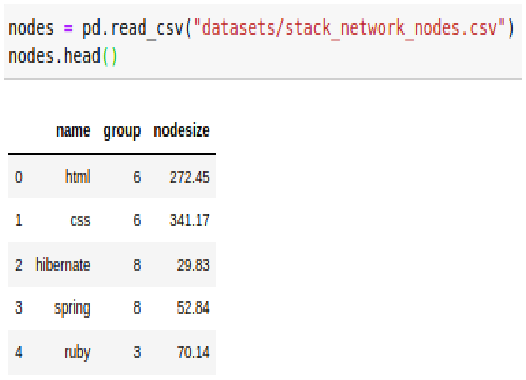
	Source	Target	Weight
0	True	True	True
1	True	True	True
2	True	True	True
3	True	True	True
4	True	True	True
...	...	...	...
485	True	True	True
486	True	True	True
487	True	True	True
488	True	True	True
489	True	True	True

490 rows × 3 columns

(b) edges.csv
It would be False if there were any NaN values

2.3.1 Nodes Table

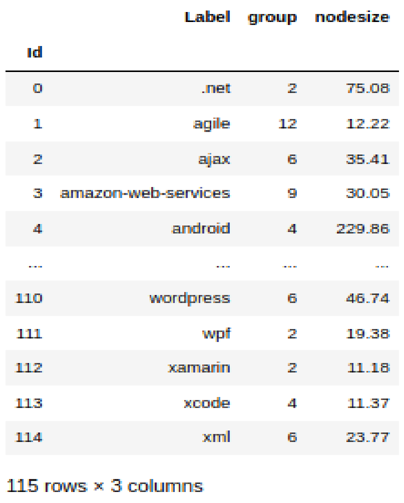
In the original dataset the Id of each tag was its own name. In order to be correctly represented as a graph in Gephi according to the tutorial, I created a unique Id for each tag.



```
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



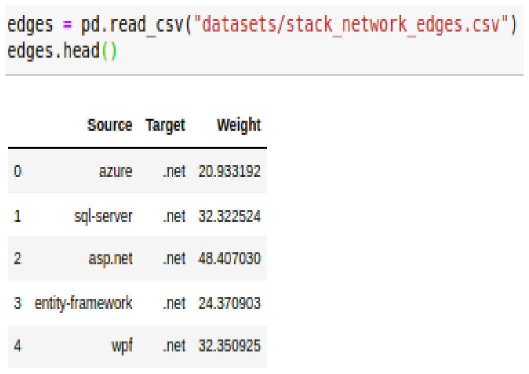
	Label	group	nodesize
Id			
0	.net	2	75.08
1	agile	12	12.22
2	ajax	6	35.41
3	amazon-web-services	9	30.05
4	android	4	229.86
...	...	...	...
110	wordpress	6	46.74
111	wpf	2	19.38
112	xamarin	2	11.18
113	xcode	4	11.37
114	xml	6	23.77

115 rows × 3 columns

(b) final nodes table

2.3.2 Edges Table

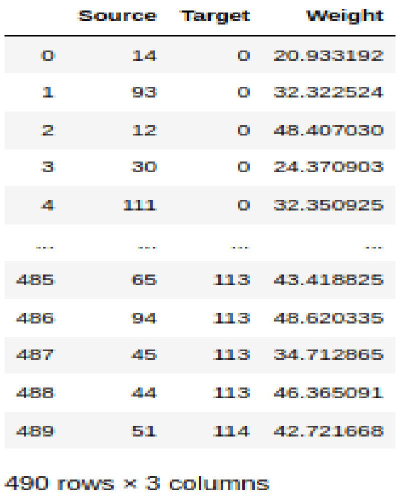
In this table, a new Id column was added, which was also the index of the table, and the column name renamed as Label.



```
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

3 Graph Representation

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

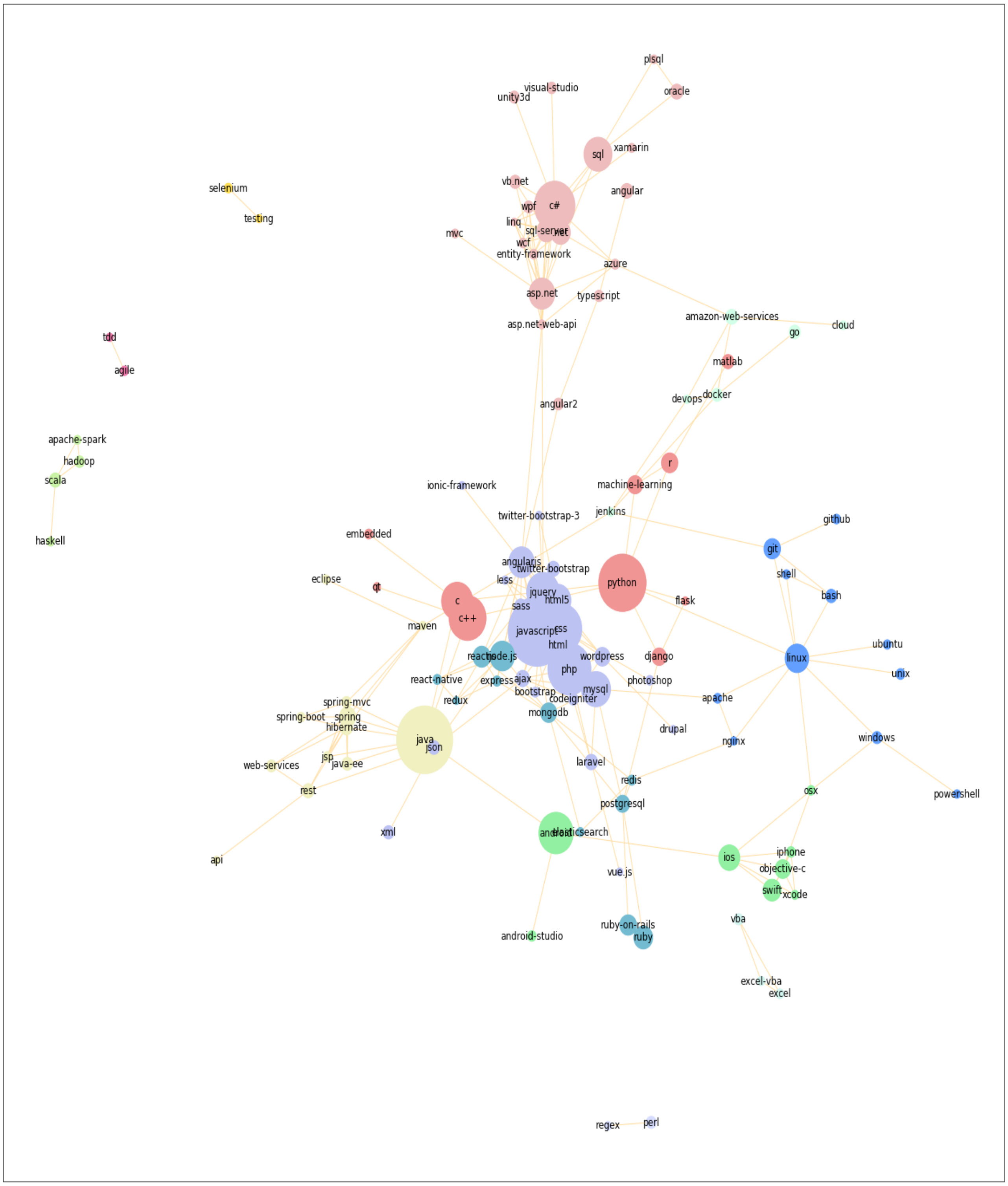


Figure 5: Network

The two following representations was made with Gephi.
The network below has different colours for each of 14 group that exist by default in the dataset.

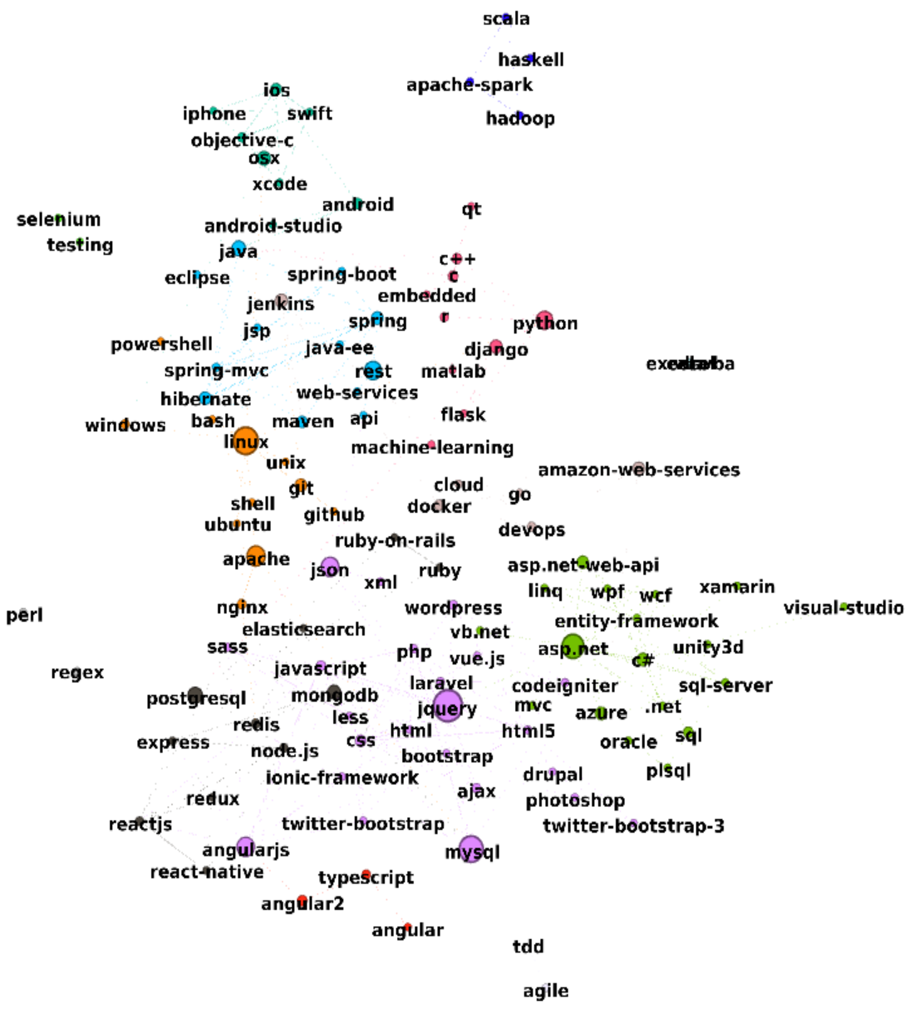


Figure 6: Network with Gephi 1st version

The network below has different different colours for each of 14 group, the edges have ranking based on their Weight, the size of nodes and their labels' font size have ranking according to the nodesize variable that exist by default in the dataset.

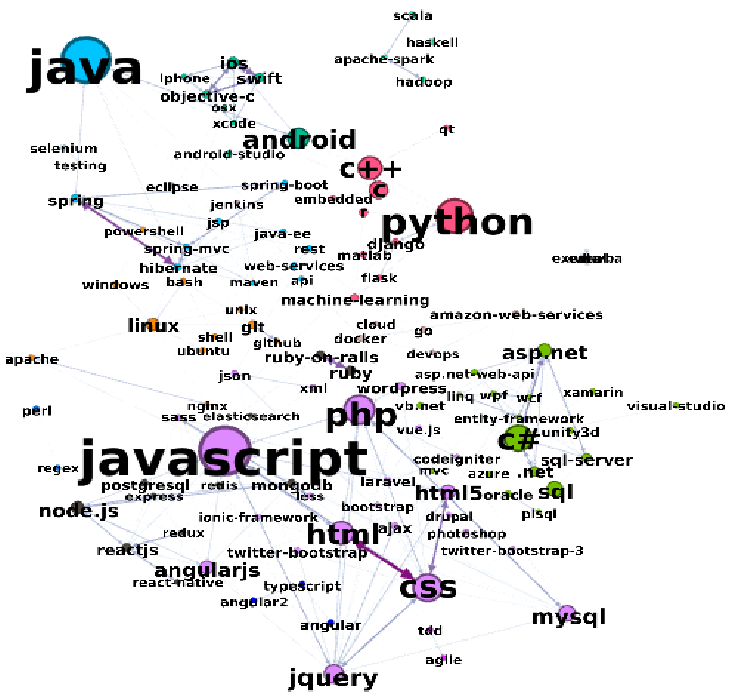


Figure 7: Network with Gephi 2nd version

4 Basic Topological Properties

- 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.

5 Component Measures

According to the Connected Components report in Gephi, for this undirected network, there are 6 weakly connected components and 6 strongly connected components. More specifically in Gephi, as Appearance Ranking value for nodes of the network we choose the Component ID and the result is the following graph.

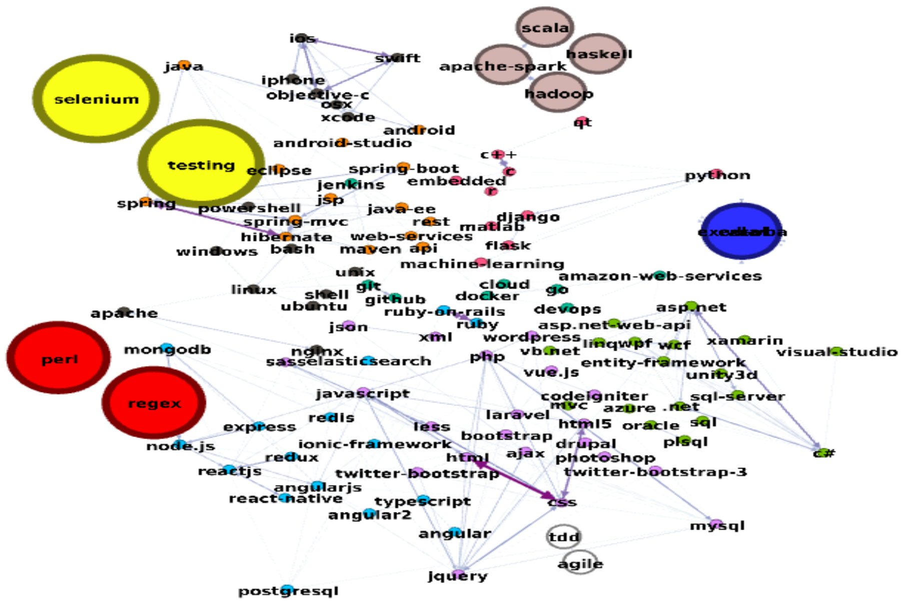


Figure 8: Components of the network

It is normal that regex and perl are one of the components, due to the fact that regural expressions are more usual in perl. Moreover, the blue circle on the right is all the excel nodes, which obviously has nothing in common with the rest nodes of the network.

6 Degree Measures

The average degree of this network is 4.2609. The tag with the highest degree is jquery with 16.

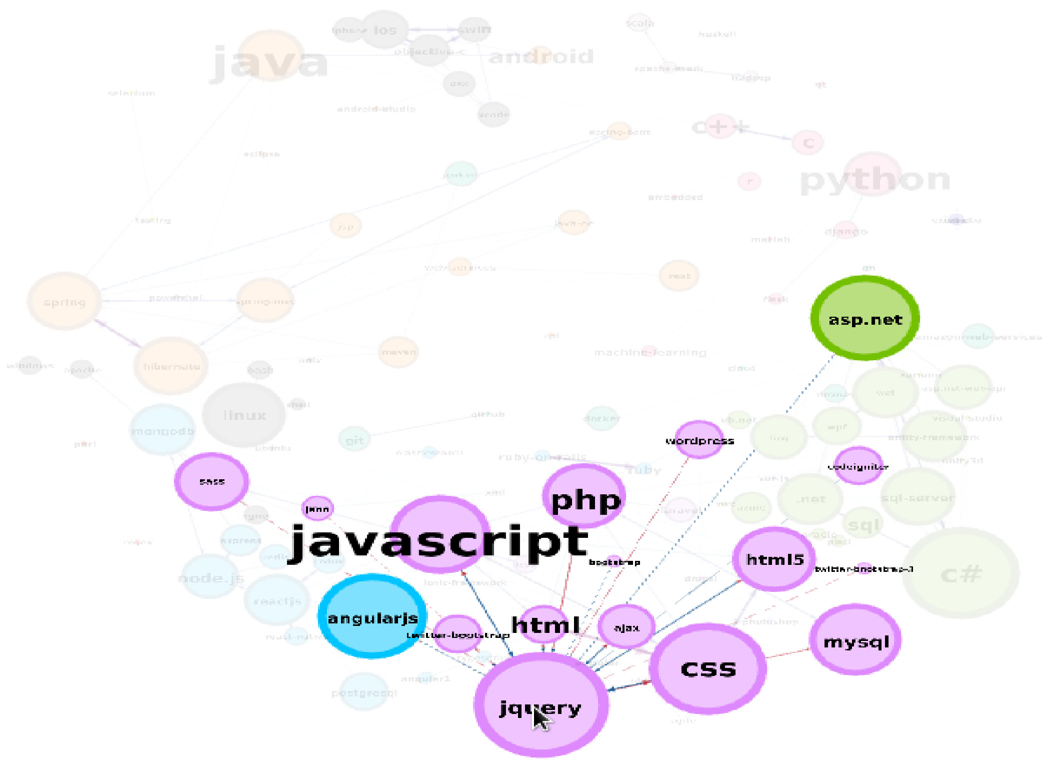


Figure 9: Node with the highest degree

The degree distribution shows us that most of the tags have degree 1 and 3.

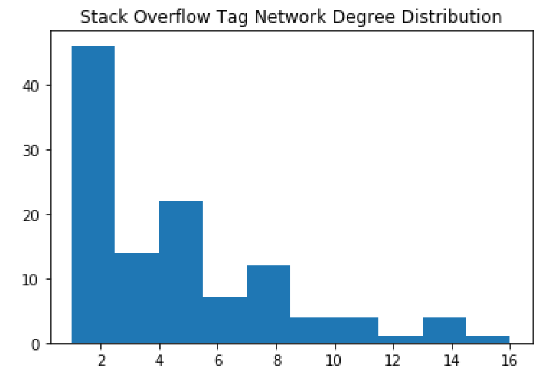


Figure 10: Node with the highest degree

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

7 Centrality Measures

7.1 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:

jquery	16
css	14
c#	14
asp.net	13
angularjs	13
javascript	12
mysql	11
html5	10
php	10
linux	10

7.2 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:

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

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.

7.3 Closeness Centrality

The Closeness Centrality shows how close a node is to the centre of th 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:

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

Top 10 tags have the highest closeness centralities degrees 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).

7.4 Eigenvector Centrality

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

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 a	0.2038

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 require jQuery.

7.5 PageRank Centrality

PageRank is a variant of Eigenvector Centrality but their main difference is that the former accounts for link direction. Each tag in the network is assigned a score based on its indegree. These links are also weighted depending on the relative score of its originating node.

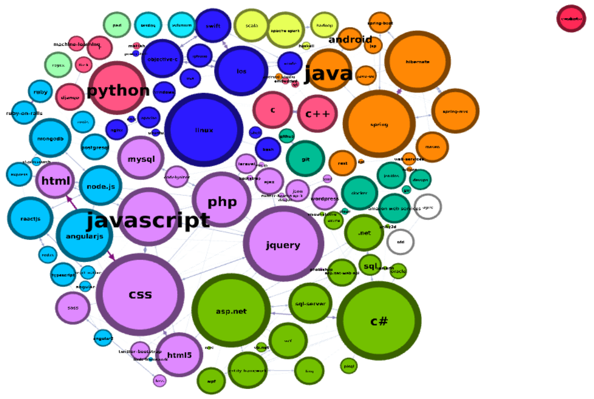


Figure 11: PageRank Centrality Network

Network with size of each node according to pagerank centrality. The colours is according to which modularity class each node belongs to and fontsize of each label is based on nodesize default variable.

8 Clustering Effects

The average clustering coefficient of this network is 0.615, which means that there is 61.5% possible that 2 tags be connected if there are connected to an A tag.

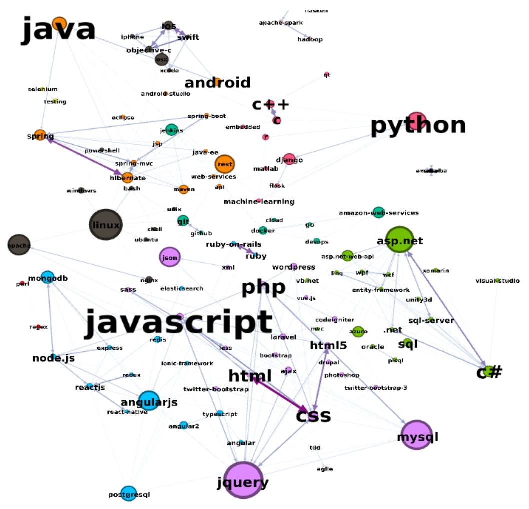
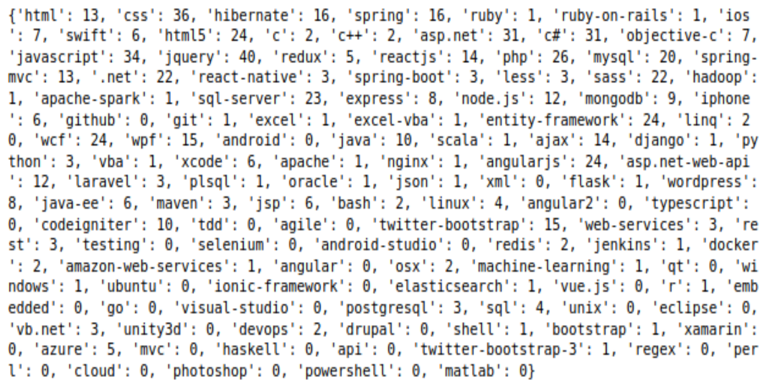


Figure 12: Cluster in Gephi

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. Triadic closure of this network is 0.4871

Also like density, transitivity is scaled from 0 to 1, and you can see that the network's transitivity is about 0.487092, somewhat higher than its 0.037376 density. Because the graph is not very dense, there are fewer possible triangles to begin with, which may result in slightly higher transitivity. That is, nodes that already have lots of connections are likely to be part of these enclosed triangles.

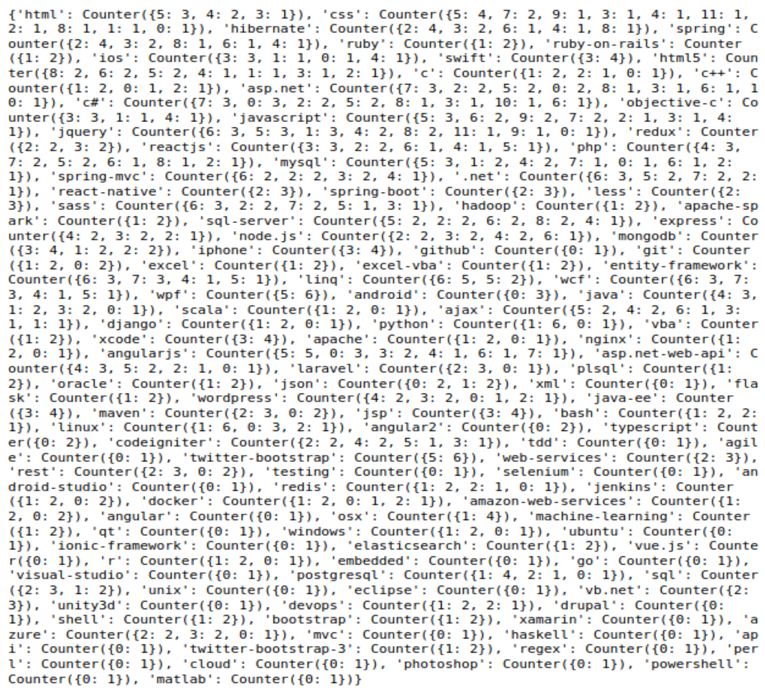
The total triangles of this network is 239.



```
{'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, 'react-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': 2
0, 'wcf': 24, 'wpf': 15, 'android': 0, 'java': 10, 'scala': 1, 'ajax': 14, 'django': 1, 'py
thon': 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, 're
st': 3, 'testing': 0, 'selenium': 0, 'android-studio': 0, 'redis': 2, 'jenkins': 1, 'docker
': 2, 'amazon-web-services': 1, 'angular': 0, 'osx': 2, 'machine-learning': 1, 'qt': 0, 'wi
ndows': 1, 'ubuntu': 0, 'ionic-framework': 0, 'elasticsearch': 1, 'vue.js': 0, 'r': 1, 'emb
edded': 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, 'haskell': 0, 'api': 0, 'twitter-bootstrap-3': 1, 'regex': 0, 'per
l': 0, 'cloud': 0, 'photoshop': 0, 'powershell': 0, 'matlab': 0}
```

Figure 13: Network Triangles Networkx

For each node, the generalized degree shows how many edges of given triangle multiplicity the node is connected to. The triangle multiplicity of an edge is the number of triangles an edge participates in.

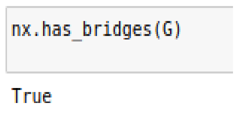


```
{'html': Counter({5: 3, 4: 2, 3: 1}), 'css': Counter({5: 4, 7: 2, 9: 1, 3: 1, 4: 1, 11: 1,
2: 1, 8: 1, 1: 1, 0: 1}), 'hibernate': Counter({2: 4, 3: 2, 6: 1, 4: 1, 8: 1}), 'spring': C
ounter({2: 4, 3: 2, 8: 1, 6: 1, 4: 1}), 'ruby': Counter({1: 2}), 'ruby-on-rails': Counter
({1: 2}), 'ios': Counter({3: 3, 1: 1, 0: 1, 4: 1}), 'swift': Counter({3: 4}), 'html5': Coun
ter({8: 2, 6: 2, 5: 2, 4: 1, 1: 1, 3: 1, 2: 1}), 'c': Counter({1: 2, 2: 1, 0: 1}), 'c++': C
ounter({1: 2, 0: 1, 2: 1}), 'asp.net': Counter({7: 3, 2: 2, 5: 2, 0: 2, 8: 1, 3: 1, 6: 1, 1
0: 1}), 'c#': Counter({7: 3, 0: 3, 2: 2, 5: 2, 8: 1, 3: 1, 10: 1, 6: 1}), 'objective-c': Co
unter({3: 3, 1: 1, 4: 1}), 'javascript': Counter({5: 3, 6: 2, 9: 2, 7: 2, 2: 1, 3: 1, 4:
1}), 'jquery': Counter({6: 3, 5: 3, 1: 3, 4: 2, 8: 2, 11: 1, 9: 1, 0: 1}), 'redux': Counter
({2: 2, 3: 2}), 'reactjs': Counter({3: 3, 2: 2, 6: 1, 4: 1, 5: 1}), 'php': Counter({4: 3,
7: 2, 5: 2, 6: 1, 8: 1, 2: 1}), 'mysql': Counter({5: 3, 1: 2, 4: 2, 7: 1, 0: 1, 6: 1, 2:
1}), 'spring-mvc': Counter({6: 2, 2: 2, 3: 2, 4: 1}), '.net': Counter({6: 3, 5: 2, 7: 2, 2:
1}), 'react-native': Counter({2: 3}), 'spring-boot': Counter({2: 3}), 'less': Counter({2:
3}), 'sass': Counter({6: 3, 2: 2, 7: 2, 5: 1, 3: 1}), 'hadoop': Counter({1: 2}), 'apache-sp
ark': Counter({1: 2}), 'sql-server': Counter({5: 2, 2: 2, 6: 2, 8: 2, 4: 1}), 'express': Co
unter({4: 2, 3: 2, 2: 1}), 'node.js': Counter({2: 2, 3: 2, 4: 2, 6: 1}), 'mongodb': Counter
({3: 4, 1: 2, 2: 2}), 'iphone': Counter({3: 4}), 'github': Counter({0: 1}), 'git': Counter
({1: 2, 0: 2}), 'excel': Counter({1: 2}), 'excel-vba': Counter({1: 2}), 'entity-framework':
Counter({6: 3, 7: 3, 4: 1, 5: 1}), 'linq': Counter({6: 5, 5: 2}), 'wcf': Counter({6: 3, 7:
3, 4: 1, 5: 1}), 'wpf': Counter({5: 6}), 'android': Counter({0: 3}), 'java': Counter({4: 3,
1: 2, 3: 2, 0: 1}), 'scala': Counter({1: 2, 0: 1}), 'ajax': Counter({5: 2, 4: 2, 6: 1, 3:
1, 1: 1}), 'django': Counter({1: 2, 0: 1}), 'python': Counter({1: 6, 0: 1}), 'vba': Counter
({1: 2}), 'xcode': Counter({3: 4}), 'apache': Counter({1: 2, 0: 1}), 'nginx': Counter({1:
2, 0: 1}), 'angularjs': Counter({5: 5, 0: 3, 3: 2, 4: 1, 6: 1, 7: 1}), 'asp.net-web-api': C
ounter({4: 3, 5: 2, 2: 1, 0: 1}), 'laravel': Counter({2: 3, 0: 1}), 'plsql': Counter({1:
2}), 'oracle': Counter({1: 2}), 'json': Counter({0: 2, 1: 2}), 'xml': Counter({0: 1}), 'fla
sk': Counter({1: 2}), 'wordpress': Counter({4: 2, 3: 2, 0: 1, 2: 1}), 'java-ee': Counter
({3: 4}), 'maven': Counter({2: 3, 0: 2}), 'jsp': Counter({3: 4}), 'bash': Counter({1: 2, 2:
1}), 'linux': Counter({1: 6, 0: 3, 2: 1}), 'angular2': Counter({0: 2}), 'typescript': Count
er({0: 2}), 'codeigniter': Counter({2: 2, 4: 2, 5: 1, 3: 1}), 'tdd': Counter({0: 1}), 'agil
e': Counter({0: 1}), 'twitter-bootstrap': Counter({5: 6}), 'web-services': Counter({2: 3}),
'rest': Counter({2: 3, 0: 2}), 'testing': Counter({0: 1}), 'selenium': Counter({0: 1}), 'an
droid-studio': Counter({0: 1}), 'redis': Counter({1: 2, 2: 1, 0: 1}), 'jenkins': Counter
({1: 2, 0: 2}), 'docker': Counter({1: 2, 0: 1, 2: 1}), 'amazon-web-services': Counter({1:
2, 0: 2}), 'angular': Counter({0: 1}), 'osx': Counter({1: 4}), 'machine-learning': Counter
({1: 2}), 'qt': Counter({0: 1}), 'windows': Counter({1: 2, 0: 1}), 'ubuntu': Counter({0:
1}), 'ionic-framework': Counter({0: 1}), 'elasticsearch': Counter({1: 2}), 'vue.js': Counte
r({0: 1}), 'r': Counter({1: 2, 0: 1}), 'embedded': Counter({0: 1}), 'go': Counter({0: 1}),
'visual-studio': Counter({0: 1}), 'postgresql': Counter({1: 4, 2: 1, 0: 1}), 'sql': Counter
({2: 3, 1: 2}), 'unix': Counter({0: 1}), 'eclipse': Counter({0: 1}), 'vb.net': Counter({2:
3}), 'unity3d': Counter({0: 1}), 'devops': Counter({1: 2, 2: 1}), 'drupal': Counter({0:
1}), 'shell': Counter({1: 2}), 'bootstrap': Counter({1: 2}), 'xamarin': Counter({0: 1}), 'a
zure': Counter({2: 2, 3: 2, 0: 1}), 'mvc': Counter({0: 1}), 'haskell': Counter({0: 1}), 'ap
i': Counter({0: 1}), 'twitter-bootstrap-3': Counter({1: 2}), 'regex': Counter({0: 1}), 'per
l': Counter({0: 1}), 'cloud': Counter({0: 1}), 'photoshop': Counter({0: 1}), 'powershell':
Counter({0: 1}), 'matlab': Counter({0: 1})}
```

Figure 14: Generaliazed Degree

9 Bridges

A bridge is a connection between two nodes that if it were to be deleted it would cause the total division of th main component into two separate components.

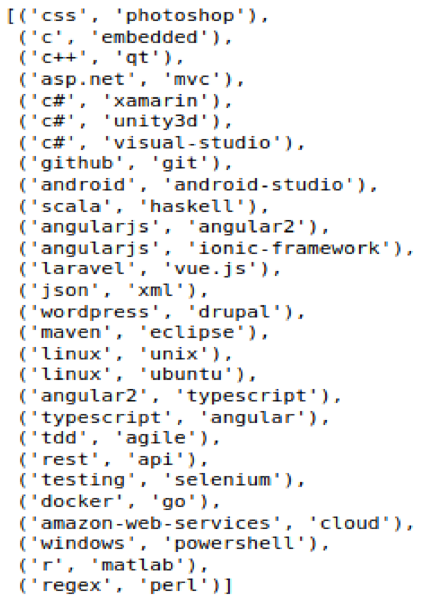


```
nx.has_bridges(G)
```

```
True
```

Figure 15: Networkx bridges

All the bridges of the network are the following:



```
[('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')]
```

Figure 16: Generaliazed Degree

10 Community

From the raw data there is the variable group, which is a result of clustering algorithm according to creators. Despite we already have an idea of communities, we run Modularity in Gephi and in networkx. This network has modularity 0.75 and we have 12 communities that correspond to a more specific prospect of groups.

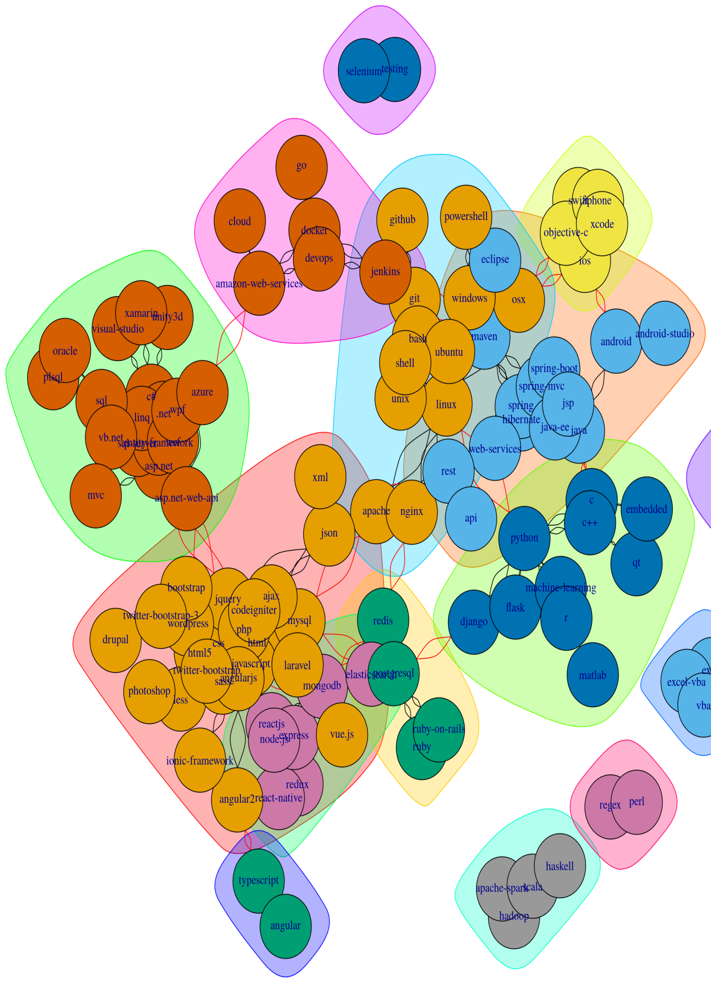


Figure 17: Communities made with igraph

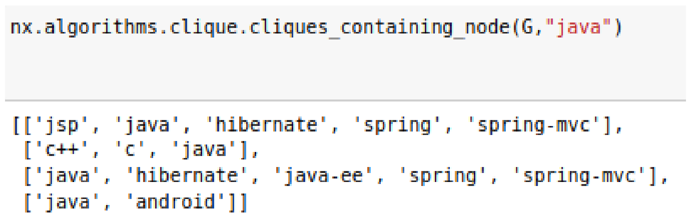
This graph is made with igraph and I found in this source.

10.1 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.

10.2 Language Specific Subgraph And Cliques

For each programming language there's a tag in the network. E.g 'java' will refer to the Java Programming language. So we can check the cliques that contains that node. We can also visualize the subgraph containing that node and all its neighbours 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 neighbours which are 1 edge away from python and also the nodes which are 2 steps away from Java.
These subgraphs 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.



```
nx.algorithms.clique.cliques_containing_node(G,"java")
```

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

Figure 18: Cliques including Java

Java participates in 4 different cliques, one for web development with jsp and sping-mvc, one for android development presumably which is connected to android. One for database management where it's adjacent to hibernate. Finally, the forth is more general acknowledgements, I presume, and Java is connected with C++ and C.

10.3 Max Cliques

By using all the functions for cliques of Networkx project 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.

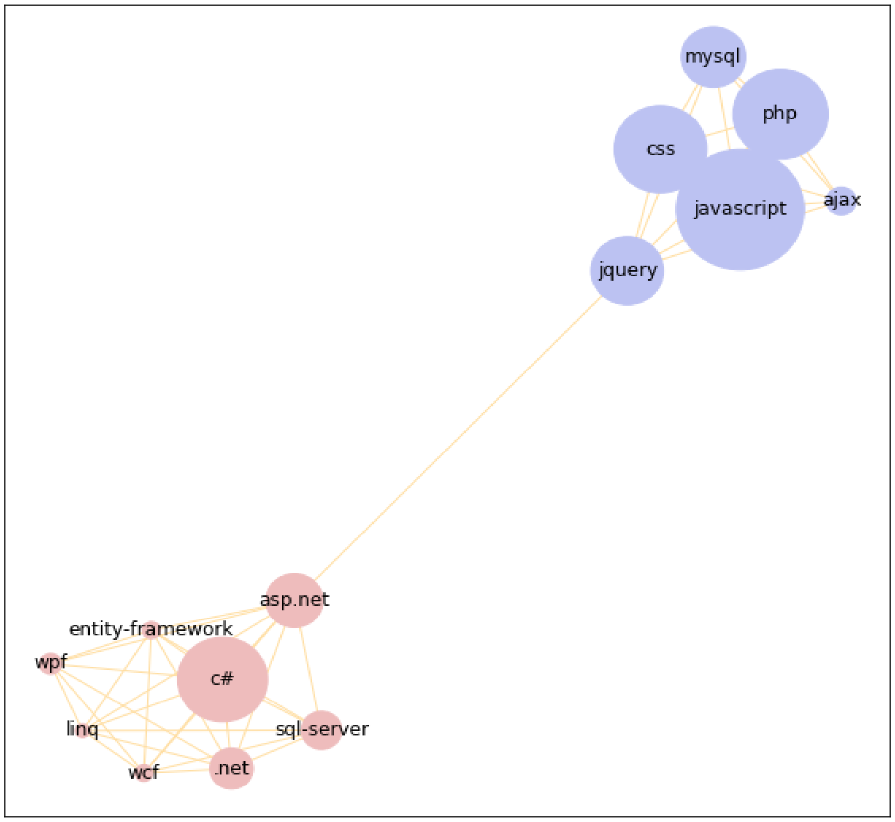
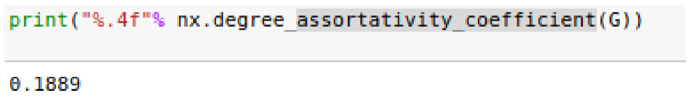


Figure 19: Max Cliques Subgraph

11 Homophily

Homophily or Assortativity is the tendency of individuals to choose friends with similar characteristic. “Like links with like.” The assortativity coefficient quantifies the extent to which connected nodes share similar properties. It is analogous to the Pearson correlation coefficient but measures the correlation between every pair of nodes that are connected. Depending on whether the node property of interest is a continuous or a categorical variable, there are two definitions of the assortativity coefficient.

The current network which concerns technologies broadly used is absolutely natural not to enhance the phenomenon of homophily, due to the role of each technology-tag in the network. A typical example of that is the Linux Operational System and Python Programming Language. There are connected, because all the applets in Linux are written in Python but Linux will never play a role of a programming language, so there is no case of choosing Python than Linux. They will always be complementary technologies.

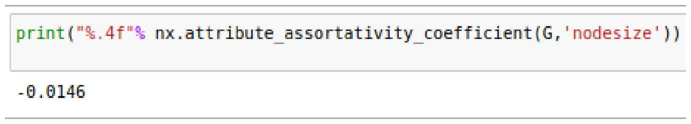


```
print("%.4f"% nx.degree_assortativity_coefficient(G))

0.1889
```

Figure 20: Network's assortativity

The assortativity coefficient based on nodes' degree is calculated to be 0.19. A value close to 0 means no strong association of the property values between connected nodes. That is to say, the tags who are friend with each other tend to have different roles and significance in technology in general.
The assortativity coefficient based on nodes' nodesize default variable is calculated to be -0.0146. A value close to 0 means no strong association of the property values between connected nodes.



```
print("%.4f"% nx.attribute_assortativity_coefficient(G,'nodesize'))

-0.0146
```

Figure 21: Network's assortativity based on nodesize

I also tried to visualize the homophily of the network by using the Axis Layout and by colouring each node according to the modularity class which belongs to. The result is the following impressing graph. If we combine the definition of homophily with the following graph, we may conclude that the nodes that are at the edges have the most negative assortativity coefficient and the other that are in the heart of that schema tend to have stronger association or recognition from the developers' community.

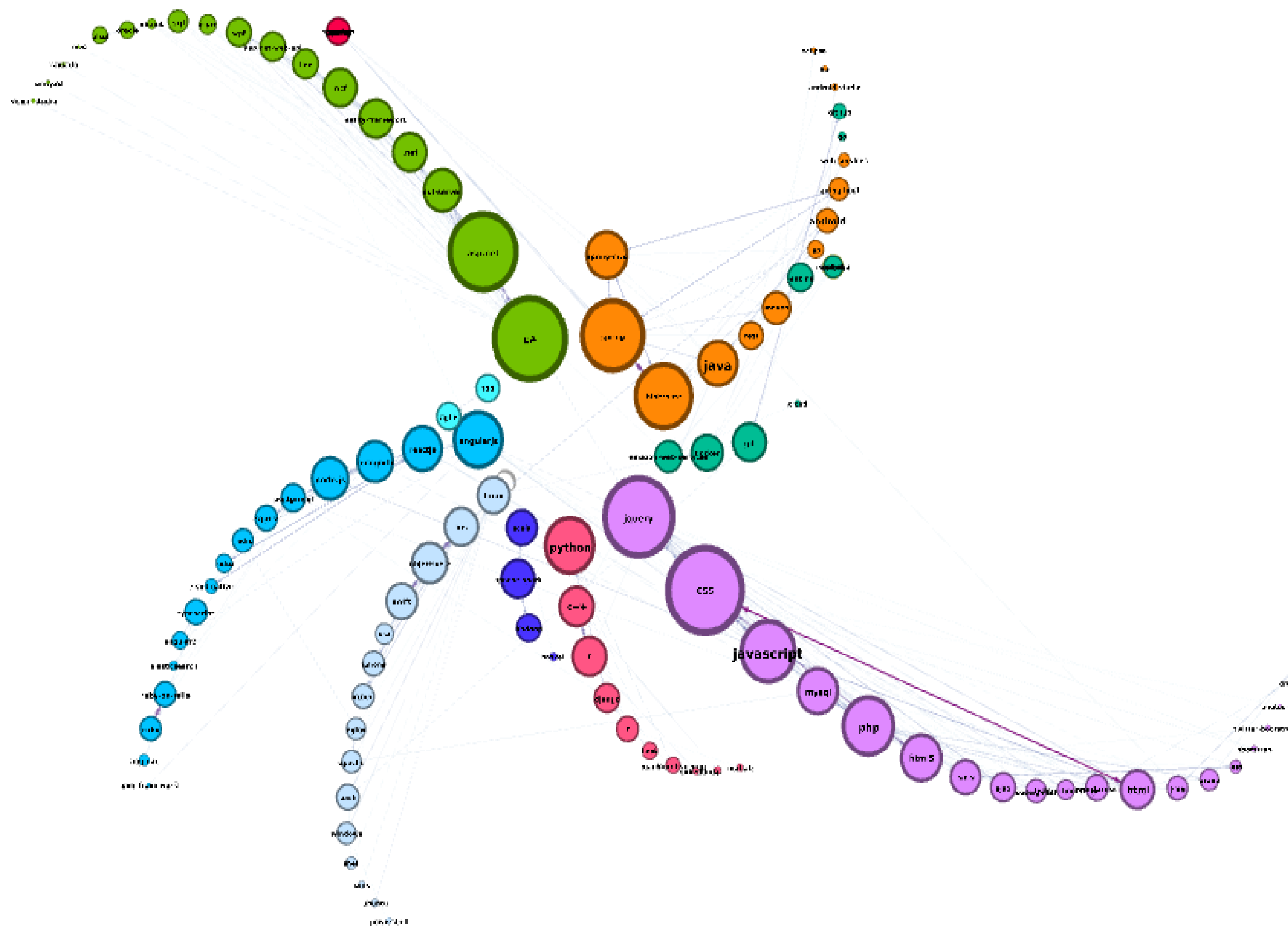


Figure 22: Network's assortativity based on modularity class

12 Trends in Stack Overflow

I decided to visualise the most loved web frameworks as a network in order to answer one part of my initial questions ”How to be up-to-date with current trending technologies? Which of them are perquisites of the others?”.
According to Stack Overflow Survey 2019 the most loved and wanted Web Frameworks are the following:

1. reactjs
2. vue.js
3. express
4. spring
5. asp.net
6. django
7. flask
8. laravel
9. angularjs
10. ruby-on-rails
11. jquery
12. drupal

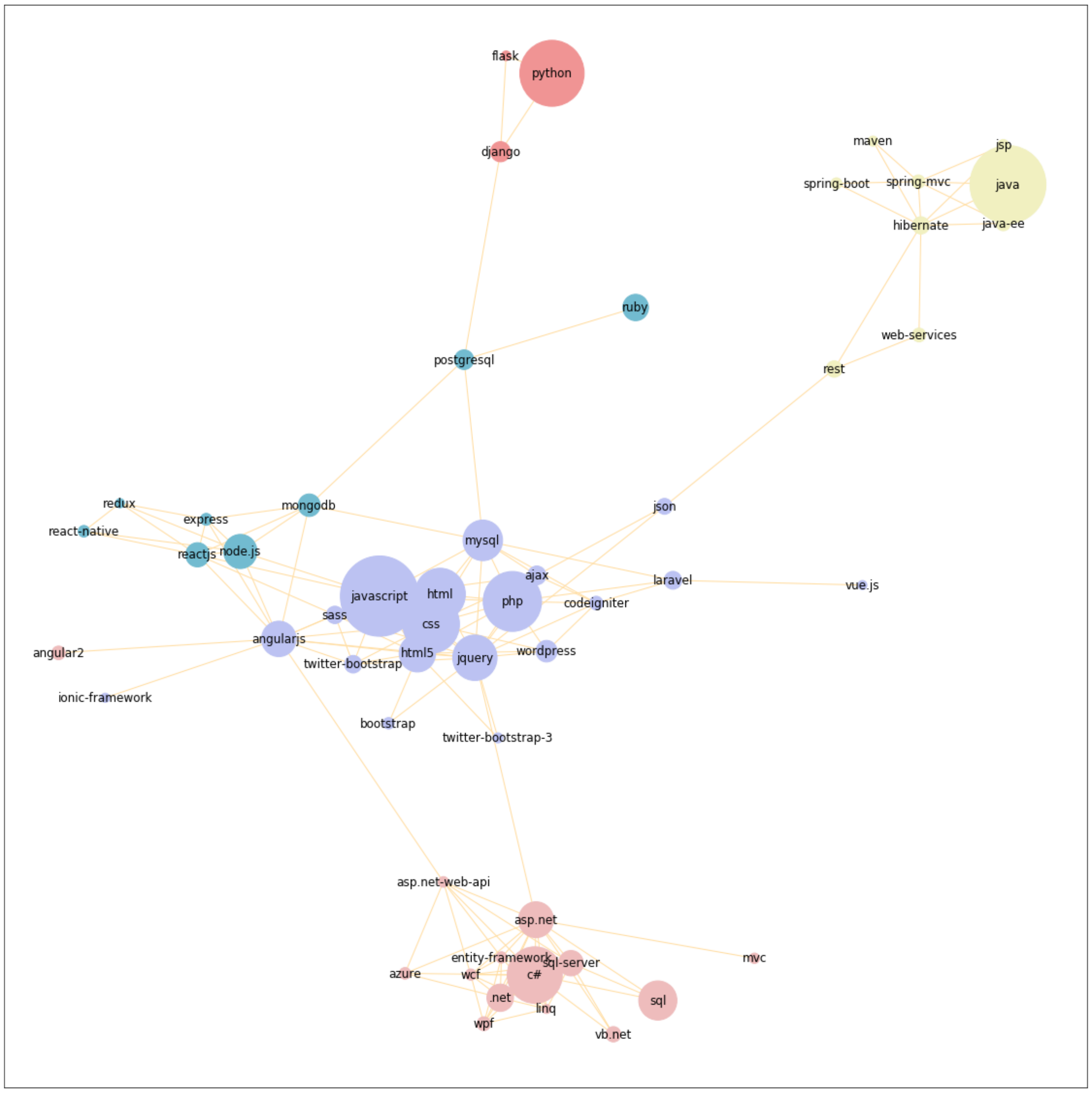


Figure 23: Most loved Web Frameworks network

As a result, I 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 neighbours (css, htl5, jquery etc.)

13 Conclusion

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). On the other hand, there are technology that are completely independent, as for instance excel. 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 2019 Trends Survey, highlighting that all widely used technologies have a lot in common as they may have a mutual starting node (e.g. Linux).

References

[1] Barabási A.L *Network Science*. Cambridge, United Kingdom: Cambridge University Press, 2012.

[2] Easley,D.,& Kleinberg, J. *Zur Elektrodynamik bewegter Körper*. (German) *Networks, crowds, and markets*. New York, NY: Cambridge University Press, 2010.

[3] Lectures' Notes Prof. D. Pournarakis,
`https://edu.dmst.aueb.gr/course/view.php?id=44`

[4] Naval Postgraduate School Monterey Homophily Presentation Prof. Ralucca Gera,
Source

[5] Networkx Documentation,
`https://networkx.github.io/documentation/stable/index.html`

[6] Gephi Documentation,
`https://gephi.org/users/`

[7] Heba Elgazzar and Adel Elmaghraby, *EVOLUTIONARY CENTRALITY AND MAXIMAL CLIQUES IN MOBILE SOCIAL NETWORKS.* https://arxiv.org/pdf/1809.02282.pdf

[8] Donglei Du, *Social Network Analysis: Centrality Measures* Source

[9] Sage researchmethods datasets *Learn About Assortativity Coefficient in Python With Data From UK Faculty Dataset* Source

[10] John Ladd, Jessica Otis, Christopher N. Warren, and Scott Weingart, *Exploring and Analyzing Network Data with Python* . Source

[11] Tutorial Repository by Mridul Seth, https://github.com/MridulS/pydata-networkx

[12] Tutorial Repository for Networkx, https://github.com/networkx

[13] Tutorial Video to import .csv in Gephi, Source