데이터마이닝 이론 및 응용 6주차 과제

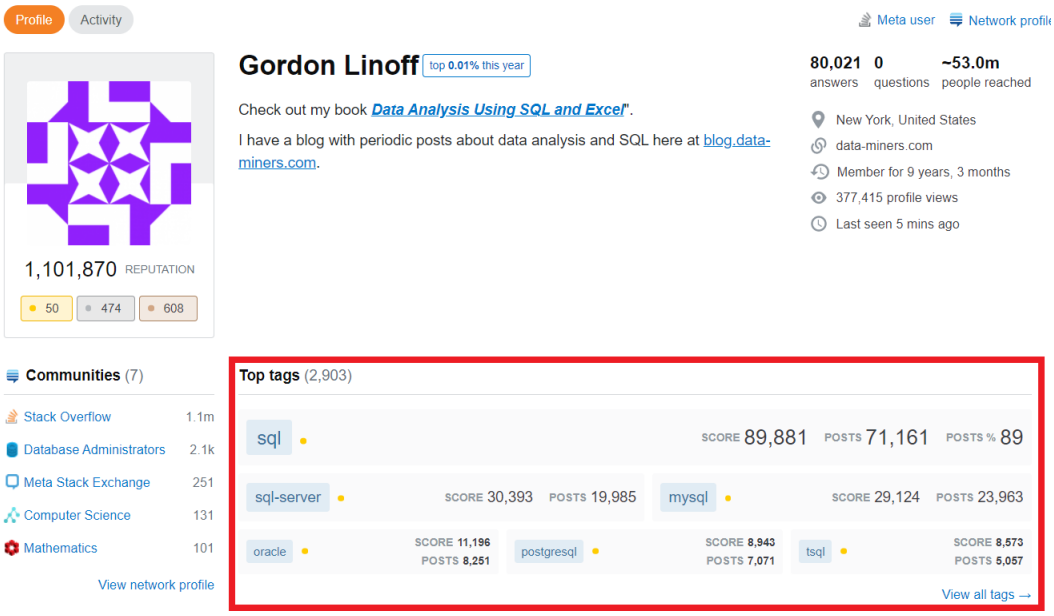
SNA

요조[곽지운, 박현준, 이정현, 최동훈]

1. Data

해당 데이터는 Kaggle의 Stackoverflow 측에서 제공한 사용자 태그 데이터이다. Stackoverflow는 컴퓨터 프로그래밍의 다양한 주제에 대한 질문들과 답변의 기능을 수행할 수 있는 웹사이트다. 각 질문들에 대해, 프로그래밍 언어이름이나 분야 같은, 그 질문의 주제와 관련된 태그들이 설정되고 수정되어질 수 있다. Stackoverflow 데이터팀에서는 많은 프로그래밍 기술들이 어떠한 관계들로 네트워크를 형성하며 생태계를 이루고있는지 파악하기 위하여 태그들 간의 관계를 이용하고자 하였다.

User가 질문글에 답변을 하는 경우 아래 사진과 같이 태그들의 score가 기록되게 되는데, Stackoverflow 데이터팀은 이러한 User데이터를 사용하여 어떤 기술 태그들이 얼마나 자주 함께 나타나는지에 관하여 정량적인 데이터화했다.



두가지 데이터를 갖을 수 있었는데, stack_network_nodes.csv와 stack_network_edges.csv 데이터를 구할 수 있었다. 이 네트워크 데이터의 구성요소는 다음과 같이 설명할 수 있다.

a. Stack_network_nodes

이 데이터에는 각 태그가 될 수 있는 프로그래밍 기술들의 이름이 적혀있고, 클러스터링을 통해 생성된 각 기술의 group이 적혀있다. 또한 해당 기술의 소속 그룹, 기술 태그 사용 빈도에 기반한 nodesize가 포함되어있다. 총 115개의 기술이 있고, **이 데이터로 노드는 각 태**

그되는 기술임을 알 수 있었다.

	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
...	...	...	...
110	perl	13	19.38
111	cloud	9	10.66
112	photoshop	6	12.62
113	powershell	5	9.85
114	matlab	1	27.21

b. Stack_network_edges

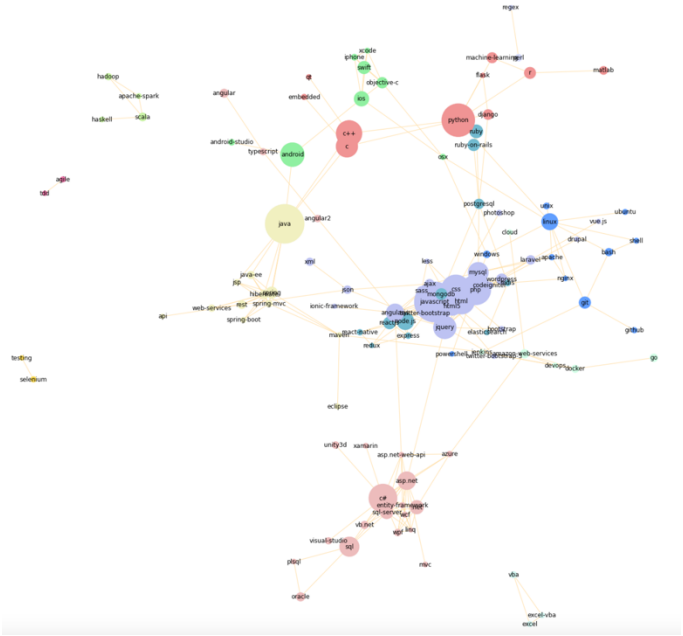
이 데이터에는 총 490개의 링크에 대한 정보가 있다. 각 노드들은 source와 target으로 표현이 되어있고, 각 링크가 얼마나 빈번하게 나타나느냐를 표현하는 value값으로 구성되어있다. **우리는 이 데이터로 링크는 두 기술들이 빈번하게 문제에 같이 등장하고 사용되는지를 나타내는 것으로 알 수 있었다..**

	source	target	value
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
...	...	...	...
485	objective-c	xcode	43.418825
486	swift	xcode	48.620335
487	iphone	xcode	34.712865
488	ios	xcode	46.365091
489	json	xml	42.721668

위의 네트워크 데이터로 분석을 진행할 것이며, 본격적인 분석 이전, Edge 데이터 내의 중복값을 확인했지만 발견되지 않았고, 방향이 없는 데이터임을 확인했다. 앞서 언급하듯이, Stackoverflow 데이터팀에서는 기술들 간의 어떠한 생태계가 이루어져 있는지 파악을 목적으로 위 데이터를 활용하고자 하였고, 우리 조는 이를 바탕으로 추가적으로 1) 학습자들이 어떠한 기술들이 연결되어서 상관관계를 이루는지 파악함으로써 기술 공부를 시작할 때 어떤 기술을 같이 공부할지 참고할 수 있고, 2) 특정 기술과 밀접한 기술의 학습을 통한 시너지효과를 분석할 수 있는 목적도 달성할 수 있다고 판단하였다.

2. Statistics

위 데이터를 통해 통계량을 구하기 전, 위 데이터의 네트워크는 노드 간 연결이 없는 경우가 생겨 오류가 발생함을 알았다. 따라서 시각화를 통해 어떤 노드가 연결이 없는지를 확인하였다.



위 그림은 네트워크를 시각화한 그림인데, 확인할 수 있듯이 연결이 안 되어있는 '별도의 네트워크'가 존재함을 알 수 있었다. 따라서 통계량을 구하기 전 각 네트워크를 구분하는 과정을 거쳤다. 총 6개의 별도의 네트워크가 나온 것으로 확인 할 수 있었고 각각 다른 네트워크 이름을 부여했다.

```
[len(c) for c in sorted(nx.connected_components(G), key=len, reverse=True)]
largest_cc = max(nx.connected_components(G), key=len)
S = [G.subgraph(c).copy() for c in nx.connected_components(G)]
S
```

```
[<networkx.classes.graph.Graph at 0x7fbe9b1611f0>,
 <networkx.classes.graph.Graph at 0x7fbe9af09160>,
 <networkx.classes.graph.Graph at 0x7fbe9883f5e0>,
 <networkx.classes.graph.Graph at 0x7fbe9883f100>,
 <networkx.classes.graph.Graph at 0x7fbe9ae5ea60>,
 <networkx.classes.graph.Graph at 0x7fbe9b1610a0>]
```

```
for i in S :
    print('Each Network info')
    print(nx.info(i))
```

	Name of Network	# of Nodes	# of Edges
1	G1	102	235
2	G2	4	4
3	G3	3	3
4	G4	2	1
5	G5	2	1
6	G6	2	1

따라서 모든 이하 진행되는 분석을 각각 네트워크 별로 진행했으며, 통계량 또한 각 네트워크 별로 도출하였고, 통합 통계량 또한 구할 수 있는 것들은 값을 내었다.

2.1. Diameter

Diameter는 네트워크에서 두 노드 간의 최단거리들 중 최댓값을 의미한다. Diameter의 경우 연결되어있지 않는 Node가 존재하는 오류로 전체 네트워크에 대한 Diameter를 구할 수 없었으므로, 각 네트워크 별로 Diameter를 구하였다.

Name of Network	# of Nodes	# of Edges	Diameter
G1	102	235	10
G2	4	4	2
G3	3	3	1
G4	2	1	1
G5	2	1	1
G6	2	1	1

102개의 Node를 갖는 제일 큰 네트워크인 G1의 경우 Diameter가 10인 것으로 확인할 수 있었다. 이는 아까 시각화를 통해 간단히 확인할 수 있었듯이, 주요한 몇개의 네트워크가 또 연결되어있는 하나의 큰 네트워크이기 때문에 최댓값의 크기가 큰 것으로 나왔다고 볼 수 있었다.

다음으로 큰 4개의 Node를 갖는 G2 네트워크의 경우에는 4개의 node이고 4개의 Edge가 있기 때문에, G1보다는 매우 낮은 2라는 Diameter를 나온 것으로 확인할 수 있었다.

다음 3개의 node인 G3와 나머지 2개의 Node를 갖는 G4,5,6모두 Diameter가 1이 나온것으로 확인할 수 있었다. 역시 노드 수가 적어 나온 것으로 확인하였고, G4,5,6의 경우 애초에 Edge가 하나씩 나왔기 때문에 Diameter의 의미가 크지 않은 것으로 생각되었다.

2.2. Density

Density는 네트워크 내에 실제 Edge수를 가능한 모든 Edge 수로 나눈 비율이다. Density의 경우에는 전체 네트워크 G에 대하여 구할 수 있기 때문에 같이 구해보았다.

Name of Network	# of Nodes	# of Edges	Diameter	Density
G	***115***	***245***	-	***0.037***
G1	102	235	10	0.046
G2	4	4	2	0.666
G3	3	3	1	1.0
G4	2	1	1	1.0
G5	2	1	1	1.0
G6	2	1	1	1.0

전체 네트워크인 G의 경우 Density가 0.037인 가장 낮은 수치로 나온 것으로 확인할 수 있었고, 네트워크 그림을 통해 역시 확인할 수 있듯이, 연결되지 않는 네트워크들이 존재하고, Node 수도 많고, 가능한 Edge가 115C2가 값인데 실제 Edge는 현저히 작기 때문에 작은 값으로 나왔다고 볼 수 있었다.

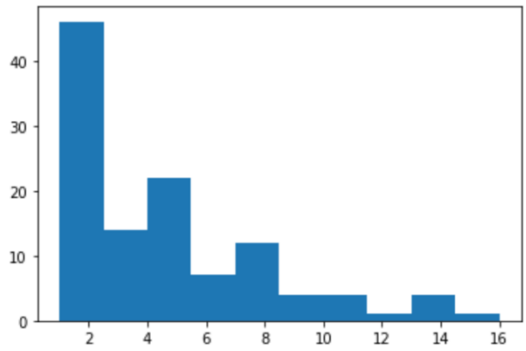
Subgraph 중 제일 큰 네트워크인 G1의 경우 Density가 0.046인 것으로 확인할 수 있었다. 이는 Diameter 때와 동일한 이유로 시각화를 통해 간단히 확인할 수 있었듯이, 주요한 몇개의 네트워크가 또 연결되어있는 하나의 큰 네트워크이기도 하고, 거의 전체 네트워크를 의미할 수 있을만큼의 Node 수와 Edge수이기 때문에 G가 낮게 나와 역시 G1도 낮게 나왔음을 알 수 있었다. 역시, 가능한 Edge 수는 102C2인데 이에 비해 현저히 낮은 Edge 수를 갖는다.

다음으로 큰 4개의 Node를 갖는 G2 네트워크의 경우에는 4개의 node이고 4개의 Edge가 있기 때문에, 가능한 총 6개 중 4개가 존재하여 0.666인 것으로 나왔다. .

다음 3개의 node인 G3와 나머지 2개의 Node를 갖는 G4,5,6모두 Density가 1이 나온것으로 확인할 수 있었다. 이는 모든 G3,4,5,6이 가능한 Edge는 모두 갖고 있다는 걸 의미하고, 실제 Edge 개수로 파악할 수 있다.

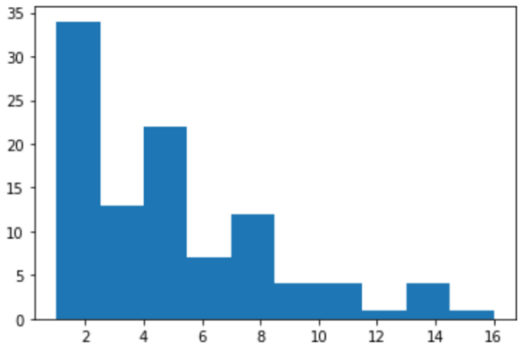
2.3. Degree

115개의 프로그래밍 기술이 각 노드를 의미하고, degree는 각 노드들에 연결된 엣지의 수를 의미한다. 다음은 전체 G에 대한 Degree Distribution을 나타낸 그래프이다.



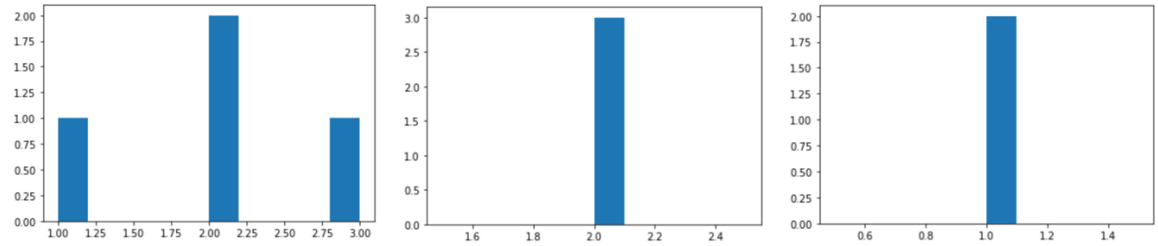

```
Name:
Type: Graph
Number of nodes: 115
Number of edges: 245
Average degree:   4.2609
```

115개의 노드와 245개의 Edge를 갖는 전체 네트워크인 G의 Degree의 평균은 5.129였으며, 1과 2의 degree값을 가지는 노드가 가장 많았고, 12,13,15,16의 값을 갖는 노드가 가장 적었다..



```
Name:
Type: Graph
Number of nodes: 102
Number of edges: 235
Average degree:   4.6078
```

다음은 개별 네트워크에 대한 설명이다. 102개의 노드와 235개의 Edge를 갖는 전체 네트워크인 G1의 Degree 평균은 4.6078로 나왔다. 앞서 언급했듯이 G1은 G의 특성과 굉장히 비슷했는데, 여기도 역시 1과 2의 degree값을 가지는 노드가 가장 많았고, 12,13,15,16의 값을 갖는 노드가 가장 적었다.



```
Name:
Type: Graph
Number of nodes: 4
Number of edges: 4
Average degree:   2.0000
```

```
Name:
Type: Graph
Number of nodes: 3
Number of edges: 3
Average degree:   2.0000
```

```
Name:
Type: Graph
Number of nodes: 2
Number of edges: 1
Average degree:   1.0000
```

위 그림은 나머지 G2,3,4,5,6에 대한 Degree Distribution이다. G,4,5,6은 모두 Node가 2개이고 그에 따라 Edge가 1개이기 때문에 평균 Degree값이 동일하게 1, 그래프도 동일하게 나왔다. G2의 경우 4개의 노드에 Edge가 4개 였는데, Degree1이 1개, Degree2가 2개, Degree3이 1개임을 알 수 있었고 평균 Degree가 2임을 알 수 있었다. G3의 경우 3개의 Node 모두 Degree 2임을 알 수 있었다.

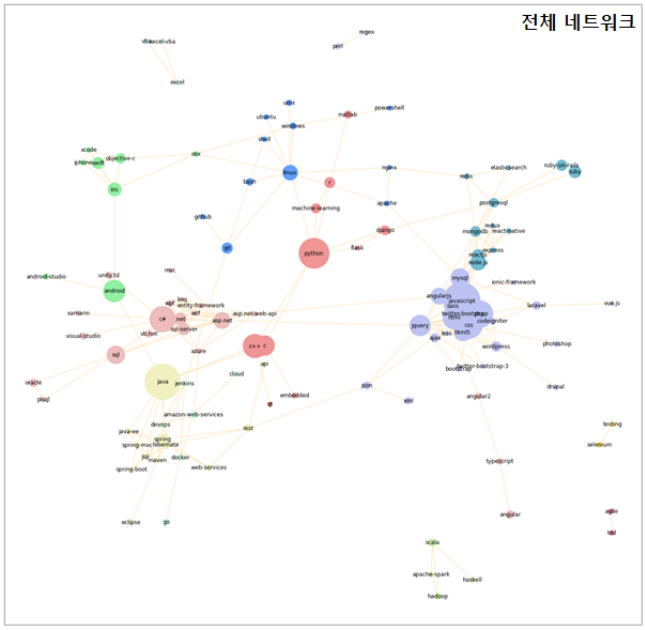
2.4. 시각화 통한 확인

위에서 구한 Diameter, Degree, Density를 네트워크 그림을 통해 확인하고자 하였다.

Name of Network	# of Nodes	# of Edges	Diameter	Density
G	***115***	***245***	-	***0.037***
G1	102	235	10	0.046
G2	4	4	2	0.666
G3	3	3	1	1.0
G4	2	1	1	1.0
G5	2	1	1	1.0
G6	2	1	1	1.0

전체 G그림을 통해 확인하였고, 이후 각각 개별적인 네트워크 그림을 그려 확인해보았다.

1) **전체 네트워크 G**

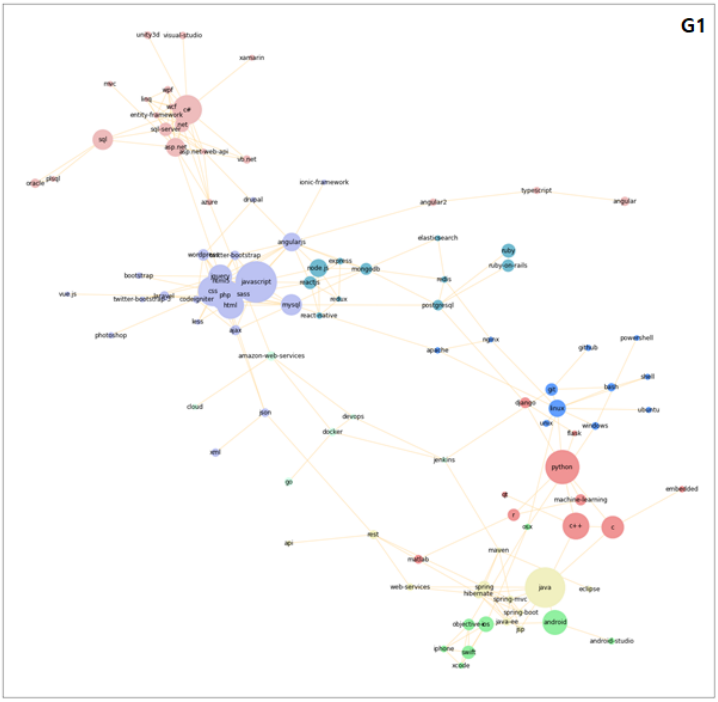


- ***Diameter :*** 전체 네트워크의 경우에는 연결되어있지 않은 Node가 존재하기에 diameter를 구할 수 없다.

- ***Density :*** 전체 네트워크의 가능한 Edge수 대비 실제 Edge수의 비율은 0.037로, 이는 전체 네트워크에서 약 3.7%의 Node pair들만이 서로간의 직접적인 연결관계를 지니고있다는 것을 의미한다. 수치가 낮은 것으로 나왔는데, 여러 서로 다른 네트워크가 있고 그림을 통해서도 낮은 Density를 확인할 수 있었다.

- ***Degree :*** 전체 네트워크에서 한 Node당 약 4.2개의 Edge를 가지고있고, 주로 1~2개와 연결된 Node들이 많고 소수의 네트워크들이 굉장히 많은 Edge를 가진 것을 네트워크 그림 상에서 찾아볼 수 있었다.

2) **G1: Node 102, Edge 235**

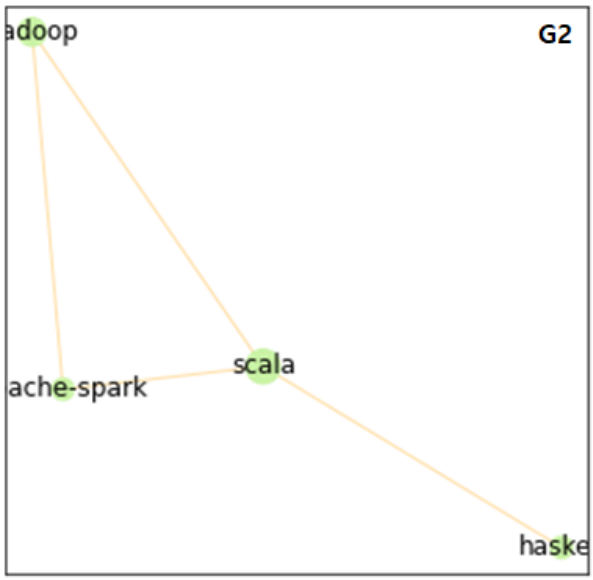


- ***Diameter :*** 네트워크 시각화 그림에서처럼 굉장히 많은 Path들을 가지고있으며, Node간의 최단거리들중 가장 큰 값이 10임을 확인해, 두 기술이 10번이하의 연결관계를 통해 직간접적으로 연결되어짐을 알 수 있었다.

- ***Density*** : G1 네트워크의 가능한 Edge수 대비 실제 Edge수의 비율은 0.046로, 이는 G1 네트워크에서 약 4.6%의 Node pair들만이 서로간의 직접적인 연결관계를 지니고있다는 것을 의미합니다. G와 마찬가지로 작은 수치가 나왔는데, 이는 여러 그룹이 서로 떨어져 있게 그림이 그려져있는 것으로도 확인할 수 있었다.

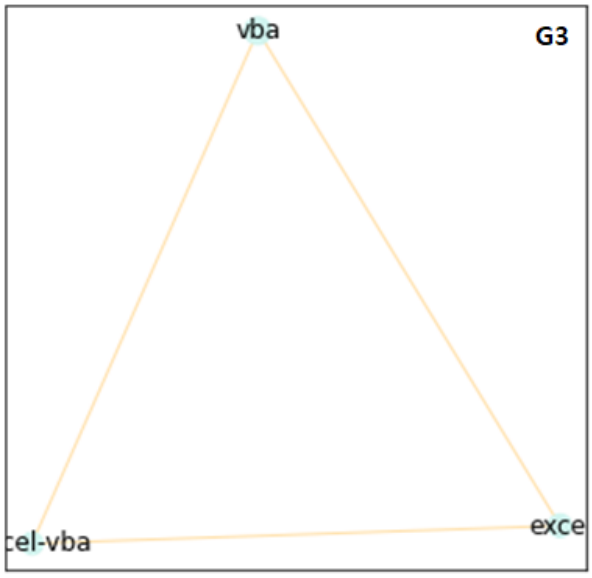
- ***Degree :*** 전체 네트워크에서 한 Node당 약 4.6개의 Edge를 가지고있고, 주로 1~2개와 연결된 Node들이 많고 소수이 네트워크들이 굉장히 많은 Edge를 가진 것을 네트워크 그림상에서 찾아볼 수 있었다.

3) **G2: Node 4, Edge 4**



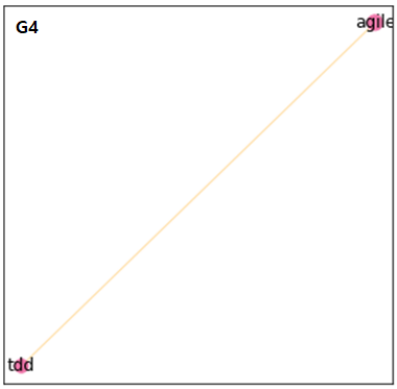
- G2네트워크는 Node간의 최단거리중 가장 큰값이 hadoop과 haskel로서 2의 diameter값을 지니고, 가능한 Edge수는 6개이고, 실제로 존재하는 Edge는 4개로서, 약 0.66의 density값을 보인다. 또한 한 개의 Node가 평균적으로 2개의 Edge들과 연결되어있기에, G2네트워크는 네트워크 상에서 Node들간에 밀접한 관련이 있고, 그것을 그림을 통해서도 확인할 수 있었다.

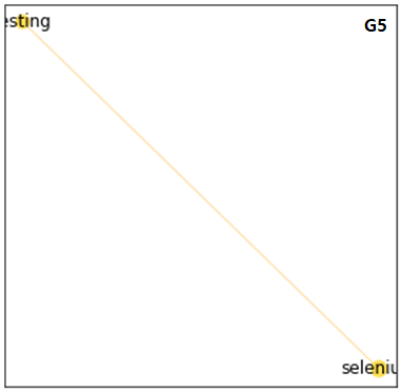
4) **G3: Node 3, Edge 3**

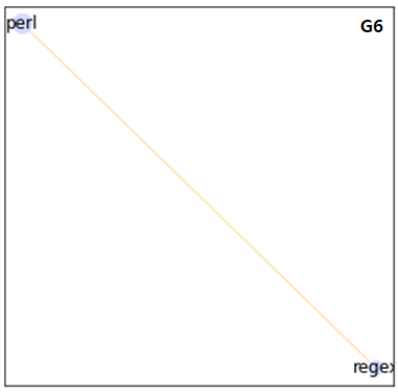


- G3네트워크는 Node간의 최단거리 중 가장 큰값이 1인 diameter값을 지니고, 가능한 Edge수는 3개이고, 실제로 존재하는 Edge는 3개로서, 1.0의 density값을 보였다. 또한 한 개의 Node가 2개의 Edge들과 연결되어있기에, G3네트워크는 네트워크 상에서 모든 Node들간에 직접적인 관련이 있고, 그것을 그림을 통해서도 확인할 수 있었다.

5) **G4,5,6: Node 2, Edge 2**.







- G4, G5, G6네트워크는 Node pair간의 거리들이 모두 1이기에 diameter값이 1이고, 가능한 Edge수는 모두 1개이고, 실제로 존재하는 Edge는 1개로, 1.0의 density값을 보였다. 또한 모든 경우에서 한 개의 Node가 1개의 Edge들과 연결되어있기에, G4, G5, G6네트워크는 네트워크 상에서 모든 Node들간에 직접적인 관련이 있고, 그것을 그림을 통해서도 확인할 수 있었다.

2.5. Centrality

Centrality에는 다양한 척도들이 존재한다. degree, betweenness, closeness, katz, PageRank, eigenvalue 등 다양한 척도들이 있다. 그 중에서 degree, betweenness, closeness, eigenvalue centrality에 따른 분석을 진행할 예정이다. 이번 데이터에서는 앞서 분석한 결과에서 본 것 처럼 Network가 총 6개 존재한다. 하지만 전체(모든) Network에 대한 Centrality 값과 Main Network의 각각에 대한 크기의 순서는 같았다. 그 이유는 다른 네트워크는 떨어져 있고, Main Network 간에 많은 노드들이 존재하기 때문이다. 그렇기 때문에 Centrality를 제일 Node가 많은 Main-Network로 진행을 했다. (아래 결과를 보면, Top5 Centrality의 순위가 같다.)

전체 Network의 Top 5 Centrality

```
degree centrality
[('jquery', 0.14035087719298245), ('css', 0.12280701754385964), ('c#', 0.12280701754385964), ('asp.net', 0.11
403508771929824), ('angularjs', 0.11403508771929824)]
--------------------
betweenness centrality
[('jquery', 0.2555399753457234), ('linux', 0.20840160874161803), ('mysql', 0.1976931477327379), ('asp.net',
0.17406690608353667), ('apache', 0.13087186063431988)]
--------------------
closeness centrality
[('jquery', 0.2895872367001647), ('mysql', 0.2778958265228288), ('ajax', 0.2586198154345401), ('css', 0.25787
451337276907), ('javascript', 0.2571334946561807)]
--------------------
eigenvector centrality
[('jquery', 0.3657638453622554), ('css', 0.338701180241117), ('javascript', 0.32563098638889276), ('html5',
0.2681052746250041), ('php', 0.26530101525817973)]
```

Main Network의 Top 5 Centrality

```
degree centrality
[('jquery', 0.15841584158415842), ('css', 0.13861386138613863), ('c#', 0.13861386138613863), ('asp.net', 0.12
871287128712872), ('angularjs', 0.12871287128712872)]
--------------------
betweenness centrality
[('jquery', 0.3259273230102584), ('linux', 0.2658049033474776), ('mysql', 0.25214684446466634), ('asp.net',
0.2220128598186257), ('apache', 0.16691993155359494)]
--------------------
closeness centrality
[('jquery', 0.3268608414239482), ('mysql', 0.3136645962732919), ('ajax', 0.29190751445086704), ('css', 0.2910
662824207493), ('javascript', 0.2902988505747127)]
--------------------
eigenvector centrality
[('jquery', 0.36576487407872865), ('css', 0.338703031202342), ('javascript', 0.3256327557015085), ('html5',
0.2681066839155206), ('php', 0.26530252590274106)]
```

1) Degree Centrality

Degree centerality는 가장 간단한 중심성 척도이다. 한 노드에 연결되어있는 Edge의 개수, 즉 앞서 분석한 Degree를 기반한 수치이고 어떤 네트워크가 가장 기본적으로 중심에 있는냐를 파악할 수 있다. 노드의 연결 정도가 그 노드의 활동성을 나타낸다. 이때 Degree에 (전체 노드수 -1)로 나누어 0~1의 값을 갖게 하여 정규화를 진행한다.

```
degree centrality
[('jquery', 0.15841584158415842), ('css', 0.13861386138613863), ('c#', 0.13861386138613863)
, ('asp.net', 0.12871287128712872), ('angularjs', 0.12871287128712872), ('javascript', 0.11
88118811881188)]
--------------------
```

jquery가 약 0.158으로 가장 높게 나왔다. 따라서 jquery 노드가 가장 다른 노드들과 연결에 영향을 많이 미친다, 즉 제일 중심에 있어보인다고 할 수 있다. 그 다음으로는 css, c#, asp.net, angularjs, javascript 순 이었다.

2) Betweenness Centrality

Betweenness centerality는 얼마나 많은 최단경로가 자신을 거쳐 가는지를 나타내며, 이질적 커뮤니티들을 얼마나 잘 이어주는지를 나태는 지표이다. 0~1사이의 값을 가지며 0은 이 노드를 지나는 최단 경로가 하나도 없음을 뜻하고 1은 모든 최단 경로가 이 노드를 지난다는 의미를 가진다. 따라서 1에 가까울수록 노드가 전체 그래프의 연결에 영향을 많이 미친다고 할 수 있다. 구하는 식을 아래와 같다.

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

σst는 노드 s와 노드 t를 지나는 모든 최단 경로의 개수를 뜻하고 σst(v)는 그 중 노드 v를 지나는 경로의 개수를 뜻한다.

```
betweenness centrality
[('jquery', 0.3259273230102584), ('linux', 0.2658049033474776), ('mysql', 0.252146844464666
34), ('asp.net', 0.2220128598186257), ('apache', 0.16691993155359494), ('json', 0.157131874
72593415)]
```

Degree Centrality 지표와 마찬가지로 jquery가 약 0.32으로 가장 높게 나왔다. 따라서 jquery 노드가 가장 다른 노드들과 연결에 영향을 많이 미친다, 즉 제일 중심에 있어보인다고 할 수 있다. 그 다음으로는 linux, mysql, asp.net, apache, json 순 이었다.

3) Closeness Centrality

Closeness centrality는 노드에서 다른 모든 노드까지 최단 경로 길이의 합에 역수를 하여 구하는 지표로 현재 노드가 얼마나 다른 노드들에 가까이 있는지를 나타내는 지표이다. 즉, 얼마나 중심적인 위치에 있는가를 나타내는 지표이며 중요한 노드일수록 다른 노드까지 거리가 짧은 것이라 이해하면 된다. Closeness centrality 구하는 식은 아래와 같다. 아래 식 처럼 길이가 짧을수록 값은 커진다.

$$Cc(A) = \frac{1}{\frac{1}{N-1}\sum_{X \neq A} l_{X,A}} = \frac{N-1}{\sum_{X \neq A} l_{X,A}}$$

Closeness Centrality의 가 큰 순서대로 Top5를 뽑은 결과이다.

```
closeness centrality
[('jquery', 0.2895872367001647), ('mysql', 0.2778958265228288), ('ajax', 0.2586198154345401), ('css', 0.25787
451337276907), ('javascript', 0.2571334946561807)]
```

다른 위의 지표와 마찬가지로 jquery가 가장 높게 나왔다. 가장 다른 노드들과 거리가 최단 경로에 위치하고 있다는 것을 알 수 있다. 그 다음으로는 mysql, ajax, css, javascript 순 이었다.

4) Eigenvector Centrality

Eigenvector centrality는 나와 연결된 노드들이 얼마나 중요한지를 나타내는 지표이다. Eignvector, eigenvalue를 구하는 식은 다음과 같다.

$$C_e(N_i) = \lambda \sum_{j}^{g} x_{ij} C_E(N_j) \; (i \neq j)$$

$C_E(N_j)$: *액터i의액텉아이겐벡터중심성*

λ : *아이겐 값*

g : *액터의 개수*

x_{ij} : *액터i와j간 연결관계의 이진값 또는 계량값*

Eigenvector Centrality의 가 큰 순서대로 Top5를 뽑은 결과이다.

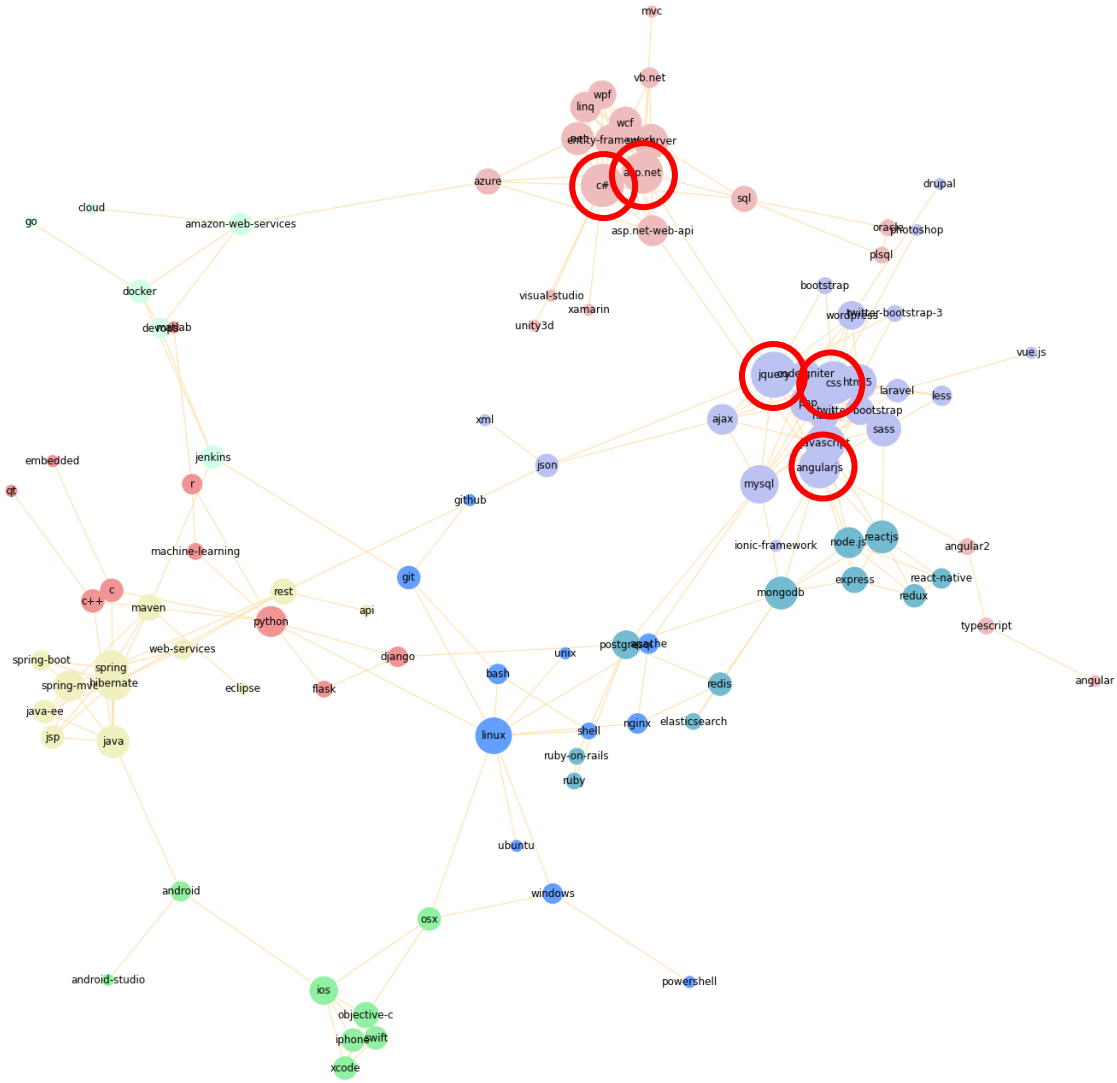
```
eigenvector centrality
[('jquery', 0.3657638453622554), ('css', 0.338701180241117), ('javascript', 0.32563098638889276), ('html5',
0.2681052746250041), ('php', 0.2653010152581797 3)]
```

jquery가 약 0.366으로 가장 크게 나왔다. 다른 중요한 노드들과 제일 많이 연결되어 있다고 볼 수 있다. 그 다음으로는 css, javascript, html5, php 순이었다.

3. Visualization

우선, 시각화에 앞서 전체 네트워크 G를 의미할 수 있을 만큼 가장 많은 node와 edge를 가진 네트워크 G1이 전체 데이터를 가장 잘 대표한다고 판단해 G1으로 시각화를 진행하였다. 각 Centrality를 node size에 반영하여 시각화를 진행해 해당 값이 특징적으로 높은 node를 추출하고 값이 가장 높은 5개의 프로그래밍 기술을 key actor로 선정했다. 다음 네트워크 그래프를 통해 각 key actor의 특성을 분석하고자 한다.

1) Degree Centrality

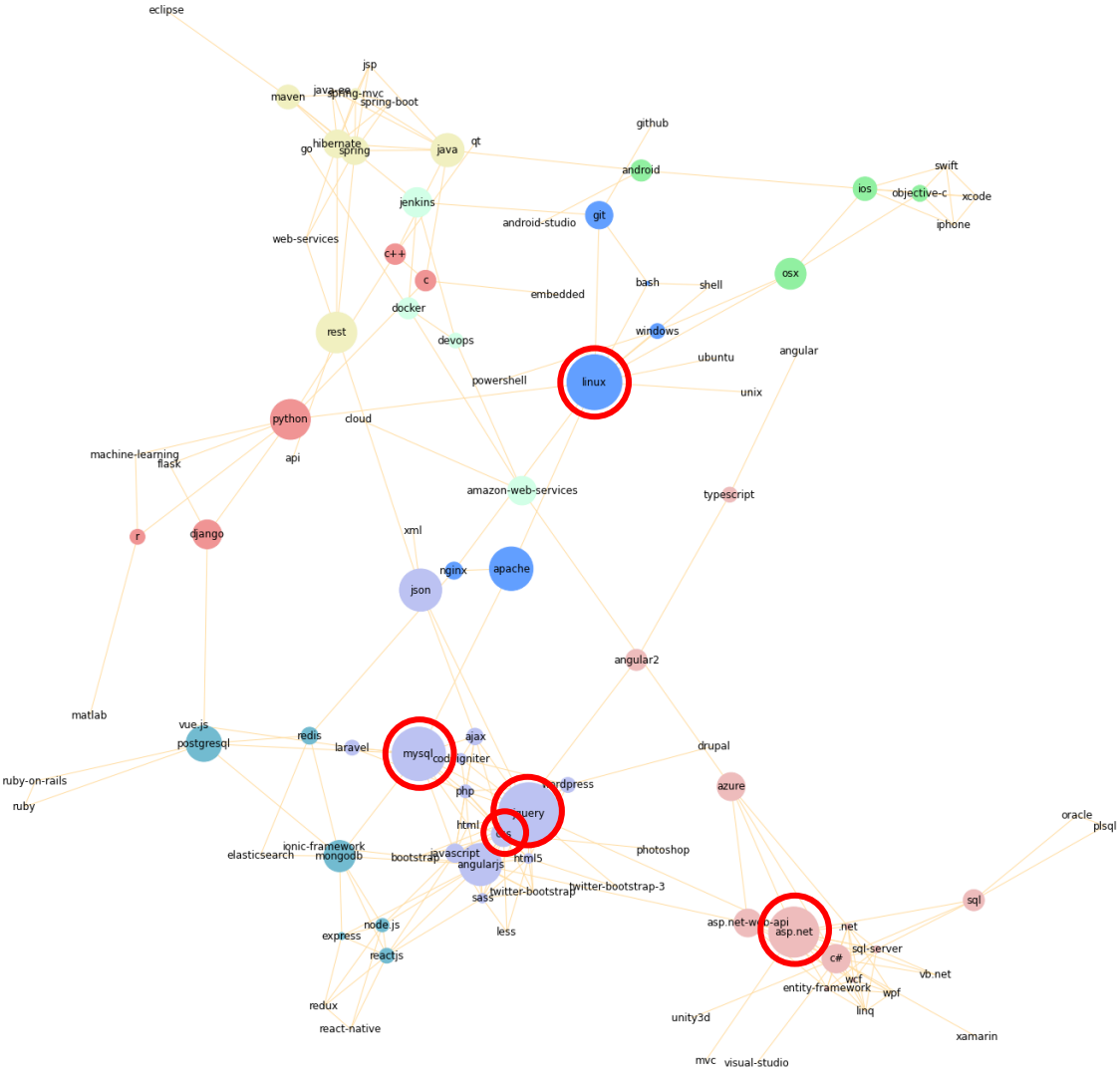


Jquery, css, c#, asp.net, angularjs이 Degree Centrality를 기준으로 가장 높은 값이 나타났고, 따라서 해당 5개의 node를 key actor로 선정했다.

같은 그룹 내에 존재하는 Jquery와 css, angularjs의 경우, 자신의 그룹 내에서 많은 다른 노드들과 연결되어 있음을 확인했다. C#과 asp.net 역시 자신의 그룹 내에서 다른 노드들과 연결되어 중심을 이루고 있음을 확인했다.

Degree가 높다는 것은 그만큼 링크가 빈번히 일어남을 의미한다. 따라서 위 다섯개의 노드는 전체적으로 링크가 빈번히 일어나 있는 노드들이고, 특히 그룹 내의 다른 노드들과 연결이 높은 것을 가시적으로 확인 할 수 있었다.

2) Betweenness Centrality



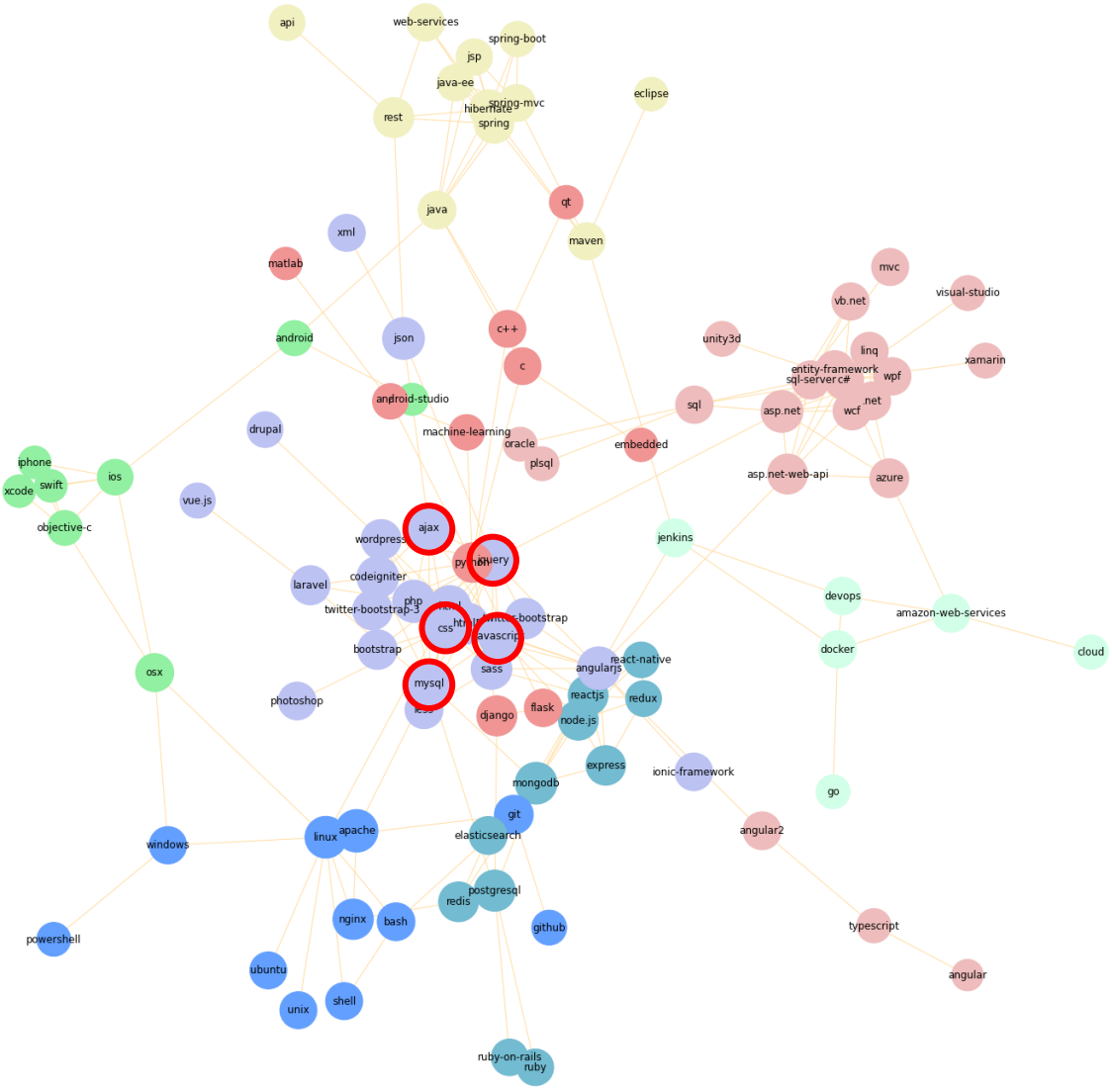
이 경우 역시 Jquery가 가장 높은 값을 나타냈고, linux와 mysql, asp.net, apache이 그 뒤를 이었다. 해당 5개의 node를 key actor로 선정했고 네트워크 그래프를 통해 이를 분석했다.

같은 그룹에 속하는 jquery, mysql, css의 경우 각각 주변의 네트워크와 연결점이 많은 것을 확인할 수 있었다. Linux의 경우 다른 네트워크들의 중간다리 역할을 하고 있음을 그래프를 통해 확인할 수 있었다. Asp.net의 경우 상대적으로 다른 노드들에 비해 다른 그룹과의 연결점은 많지 않아보였지만, jquery, mysql, css의 그룹 중 jquery와 연결점이 있음을 확인할 수 있었다.

Betweenness Centrality가 높다는 것은 그만큼 다른 그룹과의 링크가 빈번히 일어남을 의미한다. 따라서 위 다섯 개의 노드는 그림을 통해 확인할 수 있듯이 각각 다른 그룹간

의 링크가 빈번함을 확인했고, 그룹 간 링크가 많은 프로그래밍 기술이라 판단했다.

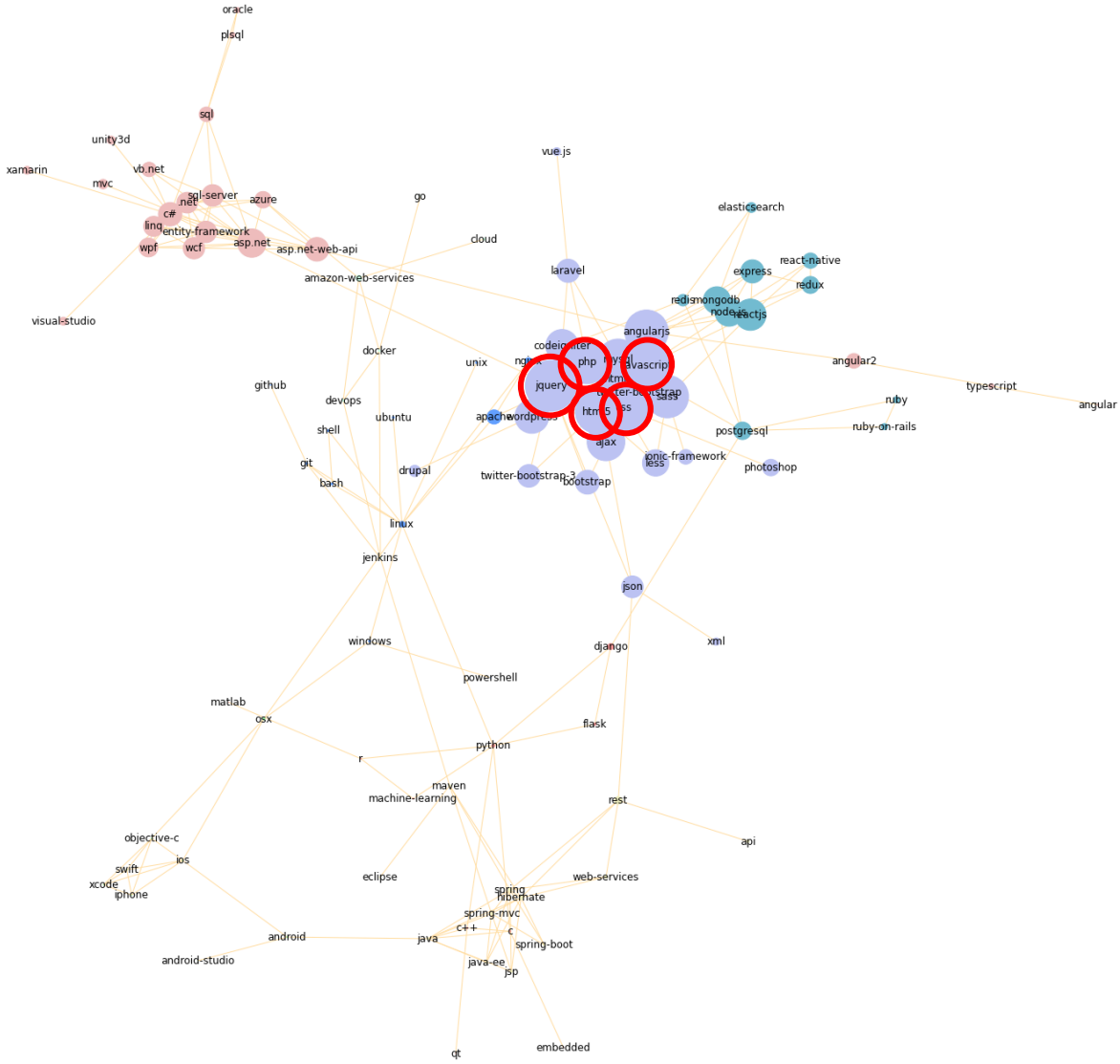
3) Closeness Centrality



Closeness Centrality를 기준으로 그래프를 그렸을 때, 가장 높은 closeness centrality 값을 가진 상위 5개는 jquery, mysql, ajax, css, javascript이다.

5개의 노드는 모두 같은 그룹에 속한 노드였다. 각 노드에서 다른 노드, 다른 네트워크로의 거리가 최단거리인 5개의 노드가 하나의 그룹에 있다는 것은 이 그룹이 전체 네트워크에서 굉장히 중심적인 위치에 있다고 판단할 수 있었고, 그림을 통해서도 가시적으로 확인할 수 있었다.

4) Eigenvector Centrality



해당 지표 역시 가장 큰 값을 가진 5개의 노드가 한 그룹에 밀집되어 나타났다. 상위 5개 jquery, css, javascript, html5, php 모두 중심부에 위치한 그룹의 노드였다.

이를 통해 해당 그룹 안의 5개의 노드와 연결된 노드들이 중요함을 알 수 있었고, 이는 5개의 노드들이 연결된 같은 그룹의 노드들, 즉 해당 그룹이 그만큼 전체 데이터에서 중요한 위치에 있음을 알려주었다.

Closeness Centrality와 Eigenvector Centrality의 네트워크 그래프 분석을 통해 jquery가 속한 네트워크가 G1의 중심부에 위치함과 동시에 중요한 노드들과 가장 많이 연결된 상당히 핵심적인 네트워크라고 결론지을 수 있었다.

4. 결론

앞서 StackOverflow의 데이터를 바탕으로 SNA 분석을 통해 다음과 같은 활용 목적을 정의했다.

1) 기술 간 상관관계 파악으로 학습 계획에 도움 2) 특정 기술과 밀접한 기술 파악을 통한 시너지 효과 창출

우리는 stackoverflow에 답변하는 user들에게 태그되는 기술들의 네트워크 분석을 통해 주로 어떠한 기술과 언어들이 관련되어있고 기술자들이 함께 알고 있는지 볼 수 있었다. 우리는 관련 있는 언어와 기술들이 서로 함께 user들이 알고 있는 것으로 보아, 훗날 기술을 공부하는 사람 입장으로서 계획을 짤 때, 어떤 기술을 추가적으로 공부할 지 참고할 수 있을 것이라 파악했다. 또한, , 네트워크 시각화 결과를 통해, 더 관련성 있는 언어 혹은 기술들 간의 그룹화를 볼 수 있었다. 이 결과로 특정 기술과 밀접한 기술 파악을 해서, 예를 들어 내가 A라는 기술을 공부하고 있는데, 어떤 추가 기술을 공부해야 시너지효과가 더 좋을 지 기술자들이 파악할 수 있게 되었다.

References

https://www.kaggle.com/stackoverflow/stack-overflow-tag-network : 데이터 출처