ECE-720 -Social Network Analysis
Assignment-1

Ravali Varanasi

#1610598

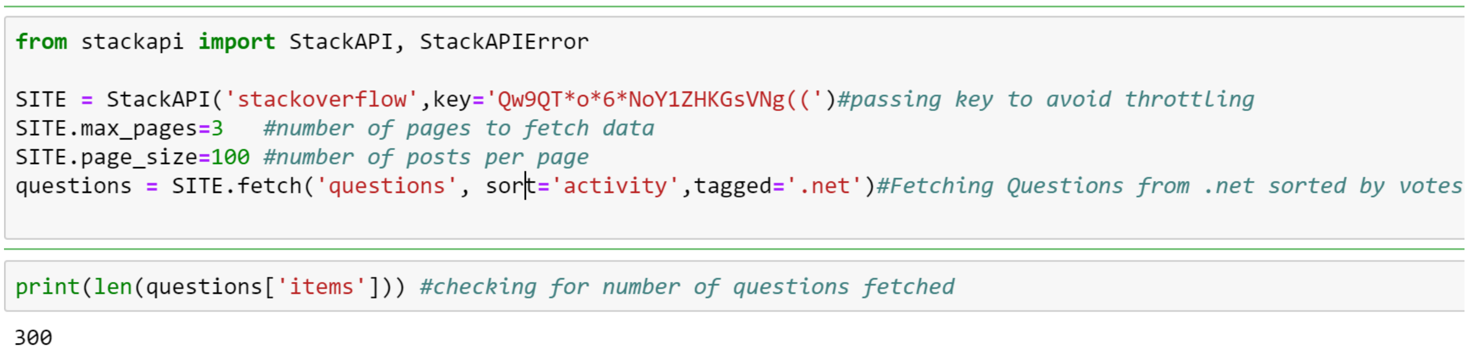
Tasks to do in this assignment:

1) Fetch Q&A from stack overflow website using StackAPI's.
2) Save all the data extracted into allposts.tsv file, (user id's, posts) into allposts-metadata.tsv and (asker id's and answerer id's) into askeranswerer.tsv file.
3) Plot network graph for askeranswerer.tsv file.
4) Find giant component of the plotted network graph and save the giant component id's in gaint_component.tsv file.

Task 01: Fetching Q&A from Stackoverflow using Stack API's:

- To extract data from stackoverflow I have used python language.
- Initially I faced issue with throttling as stackoverflow allowed only 300 requests per day, but later I have created stackapps application and received authentication.
- By using the authentication, I have increased my limit to 10000 requests per day.

Step 1:



```
from stackapi import StackAPI, StackAPIError

SITE = StackAPI('stackoverflow',key='Qw9QT*o*6*NoY1ZHKGsVNg((')#passing key to avoid throttling
SITE.max_pages=3   #number of pages to fetch data
SITE.page_size=100 #number of posts per page
questions = SITE.fetch('questions', sort='activity',tagged='.net')#Fetching Questions from .net sorted by votes
```

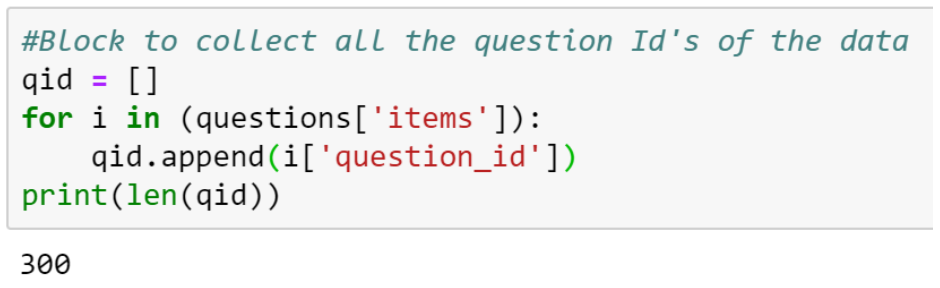
```
print(len(questions['items'])) #checking for number of questions fetched
```

```
300
```

➢ As we can see in the above code, I have used my secret key to increase requests from stackoverflow.

➢ I have fetched 300 questions which has tag ".net" and sorted by activity. To verify I have check the length of questions, which returned 300.

Step 2:

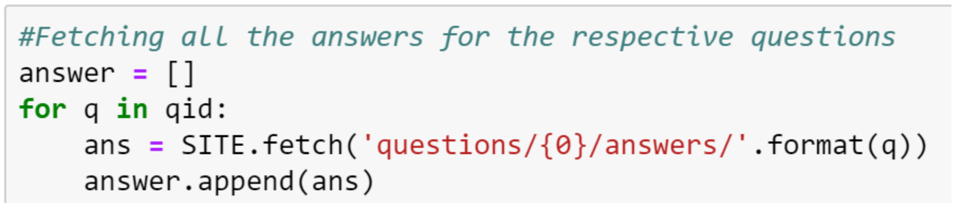


```
#Block to collect all the question Id's of the data
qid = []
for i in (questions['items']):
    qid.append(i['question_id'])
print(len(qid))
```

```
300
```

- ➢ In this step, I have collected all the question id’s to ‘qid’ array. Again to validate I checked the length which returned 300.

Step 3:



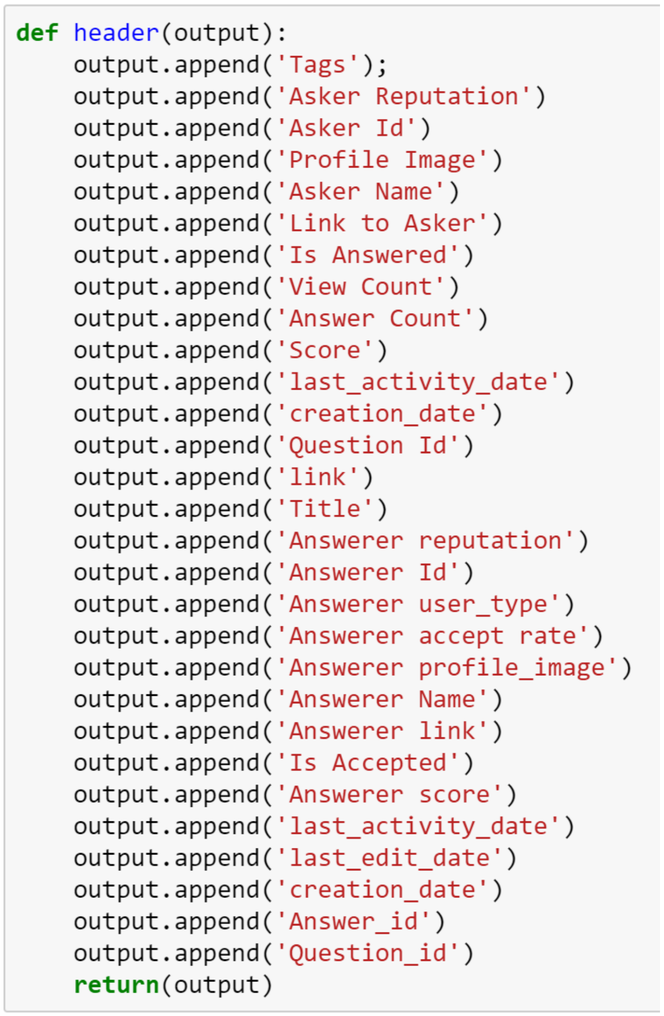
```
#Fetching all the answers for the respective questions
answer = []
for q in qid:
    ans = SITE.fetch('questions/{0}/answers/'.format(q))
    answer.append(ans)
```

- ➢ Here, I have passed all the question id’s to ‘questions/ids/answers/’ API to call all the answers for the questions fetched and appended all the answers to “answer” array.

Task 02: Save the extracted data into .TSV files:

Step 1:

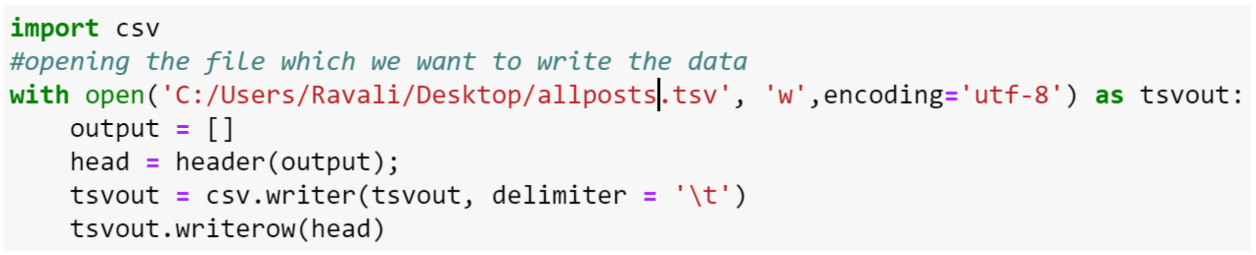
- To append both the extracted data into a single file, I am forming a header definition which has header fields of the file.



```
def header(output):
    output.append('Tags');
    output.append('Asker Reputation')
    output.append('Asker Id')
    output.append('Profile Image')
    output.append('Asker Name')
    output.append('Link to Asker')
    output.append('Is Answered')
    output.append('View Count')
    output.append('Answer Count')
    output.append('Score')
    output.append('last_activity_date')
    output.append('creation_date')
    output.append('Question Id')
    output.append('link')
    output.append('Title')
    output.append('Answerer reputation')
    output.append('Answerer Id')
    output.append('Answerer user_type')
    output.append('Answerer accept rate')
    output.append('Answerer profile_image')
    output.append('Answerer Name')
    output.append('Answerer link')
    output.append('Is Accepted')
    output.append('Answerer score')
    output.append('last_activity_date')
    output.append('last_edit_date')
    output.append('creation_date')
    output.append('Answer_id')
    output.append('Question_id')
    return(output)
```

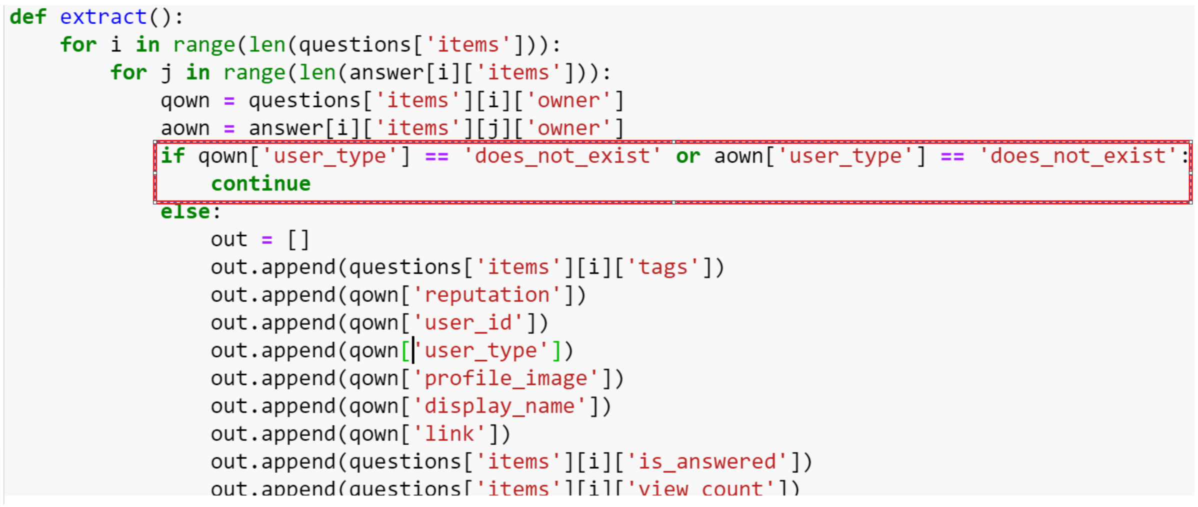
Step 2:

- In this step, we import csv to generate .tsv file. I am going to write the header in the file ‘trail1’ in my desktop directory using open command.
- I am writing the file as tab separated file (tsv) format, where fields are separated by tab space.

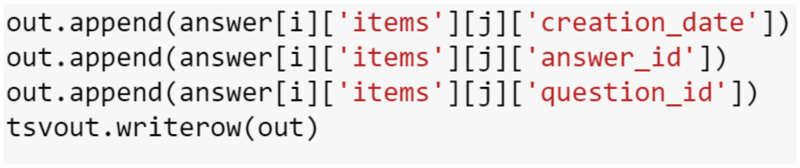

```
import csv
#opening the file which we want to write the data
with open('C:/Users/Ravali/Desktop/allposts.tsv', 'w',encoding='utf-8') as tsvout:
    output = []
    head = header(output);
    tsvout = csv.writer(tsvout, delimiter = '\t')
    tsvout.writerow(head)
```

Step 3:

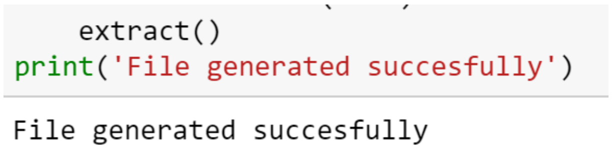
- In this step I am going to write the data fetched into the file by creating a extract definition.
- I am iterating the questions and answers and appending into the file by row wise.
- I am excluding the users who are having ‘user_type’ as not registered as marked in the below snipshot.


```
def extract():
    for i in range(len(questions['items'])):
        for j in range(len(answer[i]['items'])):
            qown = questions['items'][i]['owner']
            aown = answer[i]['items'][j]['owner']
            if qown['user_type'] == 'does_not_exist' or aown['user_type'] == 'does_not_exist':
                continue
            else:
                out = []
                out.append(questions['items'][i]['tags'])
                out.append(qown['reputation'])
                out.append(qown['user_id'])
                out.append(qown['user_type'])
                out.append(qown['profile_image'])
                out.append(qown['display_name'])
                out.append(qown['link'])
                out.append(questions['items'][i]['is_answered'])
                out.append(questions['items'][i]['view_count'])
```

- At the end of each row append I am writing into the file.


```
out.append(answer[i]['items'][j]['creation_date'])
out.append(answer[i]['items'][j]['answer_id'])
out.append(answer[i]['items'][j]['question_id'])
tsvout.writerow(out)
```

- At last I am printing a statement as shown below:


```
    extract()
print('File generated succesfully')
```

```
File generated succesfully
```

Output file:

allposts.tsv

Step 4:

- For allposts-meta_data, I am selecting only id’s of users and post. For asker-answerer data, I am selecting only asker id and answerer id. The below snip is for header of that files.In this I
- I am passing a parameter result, when result is ‘meta_data’ then it includes field along with posts or else it only takes asker id and answerer id.

```
def output(result):
    with open('C:/Users/Ravali/Desktop/trail3.tsv', 'w',encoding='utf-8') as tsvout:
        out2 = []
        out2.append('Asker Id')
        out2.append('Answerer Id')
        if result == 'meta_data':
            out2.append('Post')
        tsvout = csv.writer(tsvout, delimiter = '\t')
        tsvout.writerow(out2)
```

- Again I am iterating questions and answers as done before to write the output file.
- I am excluding the users whose ‘user_type’ is not registered.

```
for i in range(len(questions['items'])):
    for j in range(len(answer[i]['items'])):
        qown = questions['items'][i]['owner']
        aown = answer[i]['items'][j]['owner']
        if qown['user_type'] == 'does_not_exist' or aown['user_type'] == 'does_not_exist':
            continue
        else:
            out2 = []
            out2.append(qown['user_id'])
            out2.append(aown['user_id'])
            if result == 'meta_data':
                out2.append(questions['items'][i]['link'])
            tsvout.writerow(out2)
```

- Output for both meta_data and asker-answerer data is generated as shown below:

```
output('ask_ans')
output('meta_data')
```

```
File generated succesfully
File generated succesfully
```

Asker-answerer file output:

asker-answerer.tsv

Allposts_meta-data output:

allPosts-metaData.tsv

Task 03: Plotting asker-answerer network graph:

Step 1:

- Importing libraries for R-code ‘readr’ to read the file from the system, ‘igrapg’ to plot the network graph, ‘CINNA’ to find the gaint component.

```
library(readr)
library(igraph)
library(CINNA)
```

Step 2:

- Importing asker-answerer data into R.

```
#Importing the asker-answerer file generated by python code
ask_ans <- read_delim("C:/Users/Ravali/Desktop/asker-answerer.tsv",
                      "\t", escape_double = FALSE, trim_ws = TRUE)
```

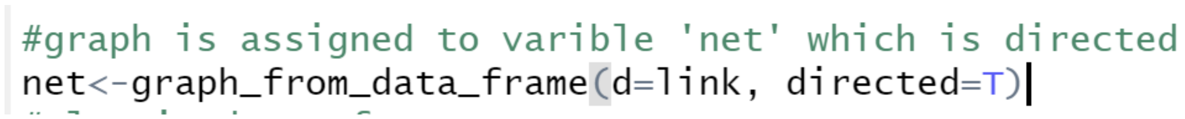
Step 4:

- Creating a data frame having two columns asker id and answerer id from the above imported data.

```
col1<-ask_ans[1];
col2<-ask_ans[2];
#Data frame is created (Link) having two coloumns askerid and answerer id
link<-data.frame(
  asker=col1,
  answerer=col2)
```

Step 5:

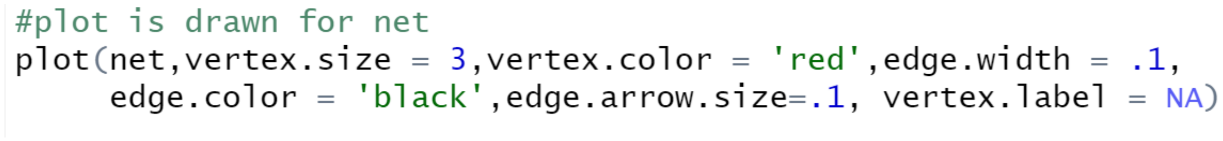
- Passing the above formed link to “graph_from_data_frame” to plot the graph ‘net’, having a directed graph.



```
#graph is assigned to varible 'net' which is directed
net<-graph_from_data_frame(d=link, directed=T)
```

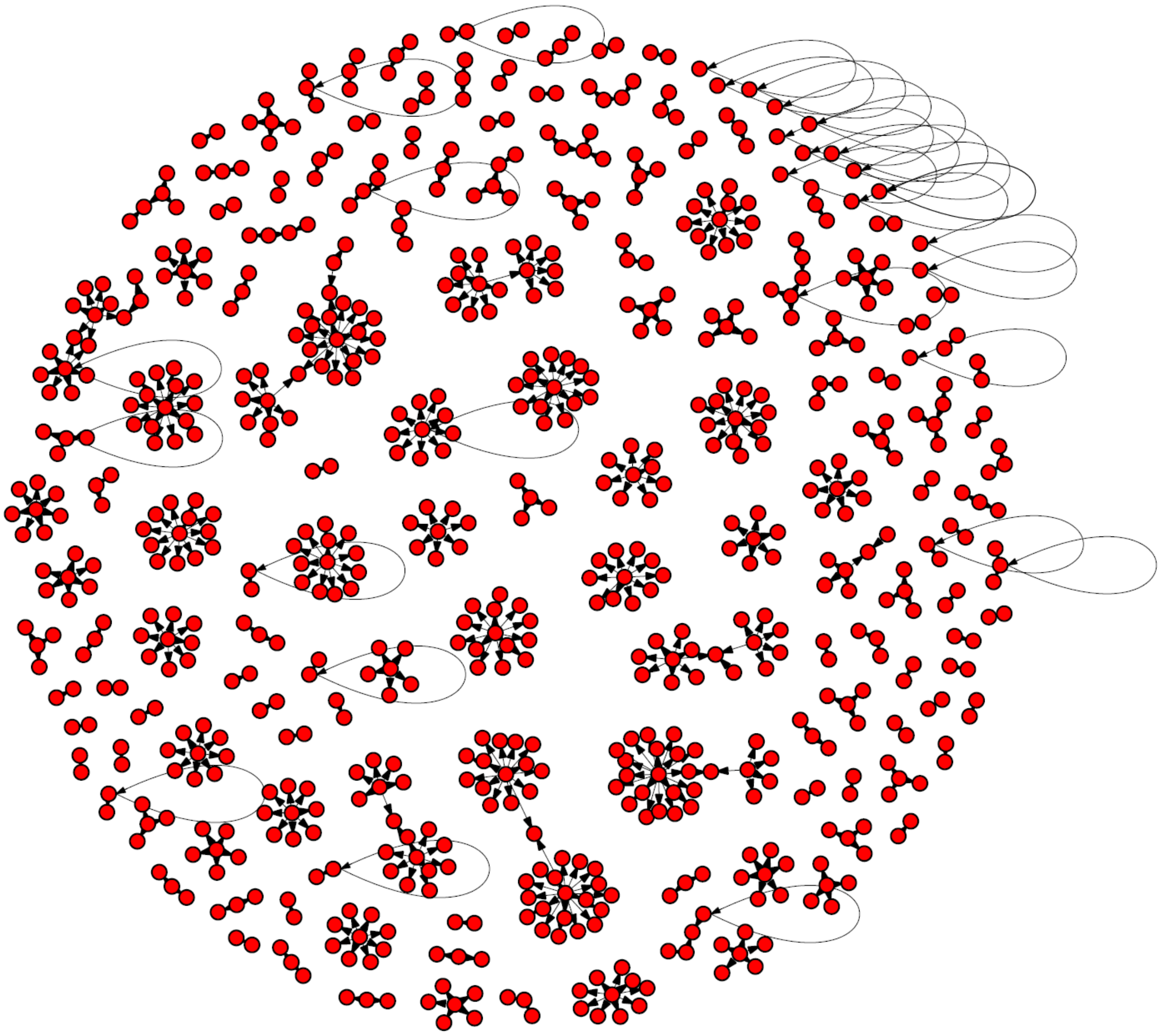
Step 6:

- In this last step I am plotting the above net, having vertex size = 3, vertex.color = ‘red', edge.width = 0.1, edge.color = 'black', edge.arrow.size=.1, vertex.label = NA.



```
#plot is drawn for net
plot(net,vertex.size = 3,vertex.color = 'red',edge.width = .1,
     edge.color = 'black',edge.arrow.size=.1, vertex.label = NA)
```

Output plot:



Task 04: Finding the giant component and exported the component data into .TSV file:

Step 1:

- Using method “giant_component_extract”, I am getting the giant component from the above plot.

```
#gaint component is extracted from the existing graph
gaint<-giant_component_extract(net,directed = TRUE)
```

Step 02:

- Again creating a new data frame ‘Link_big’ having asker id’s and answerer id’s of the giant component.

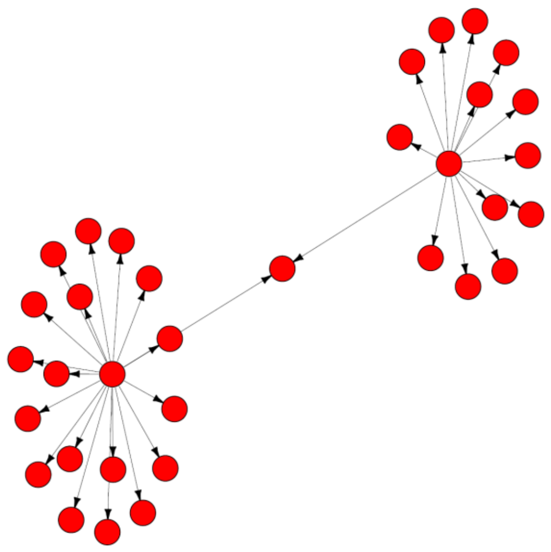
```
ask<-gaint[2][[1]][,1]
ans<-gaint[2][[1]][,2]
#gaint data is collected and formed a dataframe called 'link_big'
link_big<-data.frame(
  asker=gaint[2][[1]][,1],
  answer=gaint[2][[1]][,2]
)
```

Step 03:

- Plotting the graph as said for the previous plot but passing ‘link_big’ to net1.

```
#link_big is assigned to net1 which is directed graph
net1<-graph_from_data_frame(d=link_big, directed=T)
#Plot is done by using layout.fruchterman.reingold
plot(net1,vertex.size = 10,vertex.color= 'red',edge.width = .6,edge.color = 'black',
     edge.arrow.size=.2, vertex.label = NA,layout = layout.fruchterman.reingold)
```

Output plot:



Step 04:

- In this final step I am generating a gaint_component.tsv file which contains the data in'link_big' which is nothing but the gaint component.

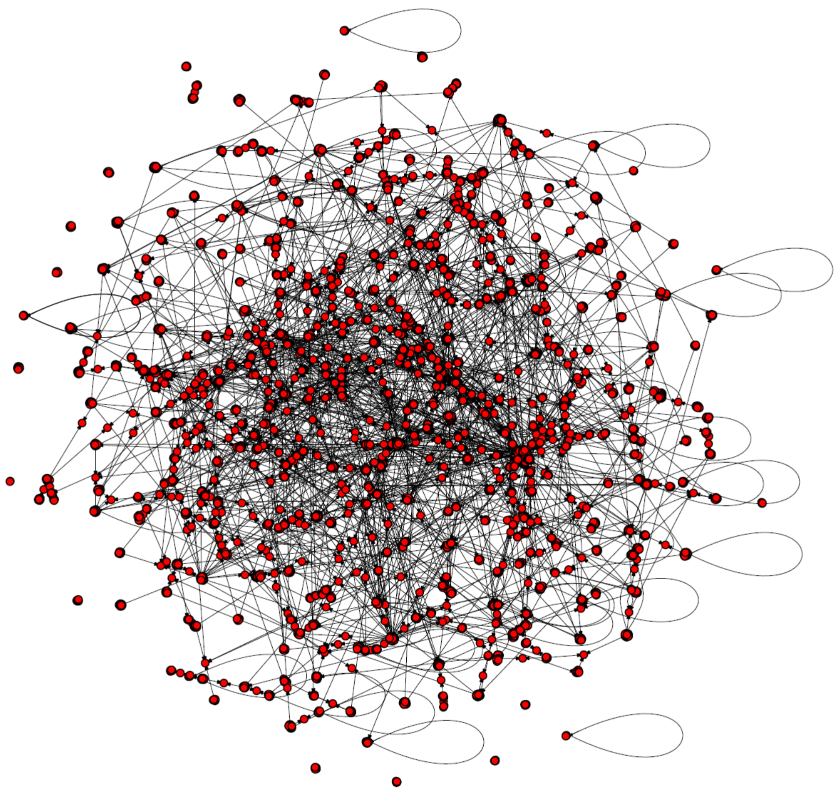
```
#Output file is generated in .tsv format for link_big
write.table(link_big, file='gaint_component.tsv', quote=FALSE, sep='\t', col.names = NA)
```

Output file:

giant_component.tsv

Problems Faced:

- Initially I was stuck with the number of API calls per day (throttling issue), I used to get just 300 posts, when I used StackApps authentication key my limit was increased to 10000 requests per day.
- I have fetched 500 questions having tag for '.net' and sorted by votes. I received almost 9000 answers for all 500 questions. To plot that huge graph, it was time consuming also graph looks too condense as shown below.



- To have better visualization and understanding I took 300 questions having tag '.net' and sorted by activity.
- Another hurdle was graphical representation of the data, I was familiar with gephi but not with R-code, later understood the representation of graphs through R-code by some online tutorials.

Google drive Link: https://drive.google.com/drive/u/0/folders/0ACOh9wpWVJUaUk9PVA

GitHub link: https://github.com/RavaliVaranasi/Stackoverflow-asker-answerer-analysis