ECE-720 -Social Network Analysis
Assignment-1

Akhil Gonna

#1610658

Goals:

1. Import question and answers from Stackoverflow using StackAPI's.
    a. Export the imported data to allposts.tsv file
    b. Export only asker id's and answerer id's to asker_answerer.tsv file
    c. Export asker id's, answerer id's and post to allposts_metadata.tsv
2. Import asker_answerer.tsv file into R-code:
    a. Plot a network graph using the data.
    b. Find the giant component(link) among the plotted graph and export the giant link data to giant_component.tsv file.

Goal 1;

To import question and answers from Stackoverflow using StackAPI's:

- I have used python to import the data and export the data.
- There some certain limits for questing data from stackoverflow. For a single IP address we can request only 300 per day.
- As I am fetching above 300 questions and posts, I have used my authentication by creating an application in StackApps.
- By using the authentication, I have increased my limit to 10000 requests per day.

Step 1:

- To import data we need to install StackAPI library into python.

```
from stackapi import StackAPI
```

Step 2:

- I have passed stackoverflow web link and my authentication secret key into 'SITE'.
- I have requested for 5 pages, where each page includes 100 questions.

```
#passing website and key
SITE = StackAPI('stackoverflow',key='1m2D1EmsS*nKHGMwKBEAgQ((')
SITE.max_pages=5
SITE.page_size=100
```

Step 3:

- Fetching questions which has tag ‘python’ and sorted by activity by using API “questions/” and also displaying number of questions fetched.

```
#calling questions API to get questions wtagged with python and sorted by activity
questions = SITE.fetch('questions', sort='activity',tagged='python')
print("Number of questions fetched: ",len(questions['items']))
```

Result for above code is:

```
Number of questions fetched:  500
```

Step 4:

- Passing all the question id’s into a an array “question_id” which is used to fetch answers.

```
#passing all question ids into a array
question_id = []
for i in (questions['items']):
    question_id.append(i['question_id'])
print("All question id's are appended to an array 'question_id'")
```

Result:

```
All question id's are appended to an array 'question_id'
```

Step 5:

- By passing all the above question id’s to API “questions/ids/answers/” I am fetching all the answers for the questions and appending to an array “answers”.

```
answers = []
#calling API to get all answers of the questions by passing question ids
for i in question_id:
    answers.append(SITE.fetch('questions/{0}/answers/'.format(i)))
print("Answers for all the questions are loaded")
```

Result:

```
Answers for all the questions are loaded
```

Step 6:

- In this step I am passing file names as per requirement into the main function:

```
main('allposts')
main('meta_data')
main('ask_ans')
```

Step 7:

- In the main function I am opening the file and we are writing the file name as per the requirement and passing the file name to fetch function.

```
def main(file):
    import csv
    #opening and writing the fetched data into out.tsv file
    if file == 'allposts':
        with open('C:/Users/Akhil/Desktop/allposts.tsv', 'w',encoding='utf-8') as tsvout:
            fetch(file,tsvout)
    elif (file == 'meta_data'):
        with open('C:/Users/Akhil/Desktop/allposts_metadata.tsv', 'w',encoding='utf-8') as tsvout:
            fetch(file,tsvout)
    elif (file == 'ask_ans'):
        with open('C:/Users/Akhil/Desktop/ansker_answerer.tsv', 'w',encoding='utf-8') as tsvout:
            fetch(file,tsvout)
```

Step 8:

- In Fetch() function: we are calling header function and writing the header into the file where each field is separated by '\t' which is tab spaced.

```
def fetch(file,tsvout):
    result = header(file);
    tsvout = csv.writer(tsvout, delimiter = '\t')
    #writing header
    tsvout.writerow(result)
    extract(file,tsvout)
```

- In header function I am appending the header name based on the file name and returning to the result as shown in the below snapshot.

```
def header(file):
    result = []
    if (file == 'allposts'):
        result.append('Tags'); result
        result.append('Profile Image'
        result.append('Is Answered');
        result.append('Score'); resul
        result.append('last_edit_date

        result.append('Answerer reput
        result.append('Answerer accep
        result.append('Answerer link'
        result.append('last_activity_
        result.append('Answer_id'); r
    elif (file == 'meta_data'):
        result.append('Asker Id');resu
    #header for ask_ans file
    elif (file == 'ask_ans'):
            result.append('Asker Id')
    return(result)
```

- I am also calling extract function by passing file and tsvout variables to write the fetched data.

Step 9:

- In extract function I am iterating ‘questions’ and ‘answers’ to write the output file and I am excluding the data where ‘user_type’ is not registered.

```
def extract(file,tsvout):
    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 = []
                if (file == 'allposts'):
                    out.append(questions['items'][i]['tags']);out.append(qown['reputation'])
                    out.append(qown['user_id']); out.append(qown['user_type'])
```

- If any of the field is not available in the fetched data, I am manually writing ‘N/A’ as shown below:

```
if 'last_edit_date' not in questions['items'][i]:
    out.append('N/A')
else:
    out.append(questions['items'][i]['last_edit_date'])
```

- At the end of each iteration, I am writing each record into the file.

```
 elif (file == 'meta_data'
     out.append(qown['user

 tsvout.writerow(out)
```

Step 10:

- At last in the main function I am printing the file name generated**.**

```
 print('{0}.tsv file generated'.format(file))
```

Output:

```
main('allposts')
main('meta_data')
main('ask_ans')
```

```
allposts.tsv file generated
meta_data.tsv file generated
ask_ans.tsv file generated
```

Files Generated:

a. **All posts file:**

b. **All posts-metadata file:**

c. **Asker-answerer file:**

Goal 2:

Plotting asker-answerer network graph:

Step 1:

- Importing library ‘readr’ to read the file from the system.
- Imported data is assigned to a variable ‘link’

```
library(readr)
#Reading the file generated by pyhton code
link <- read_delim("C:/Users/Akhil/Desktop/asker_answerer.tsv",
                   "\t", escape_double = FALSE, trim_ws = TRUE)
```

Step 2:

- Importing library ‘igraph’ to plot the network.
- The above imported link is sent to the network which is indicating as directed network.

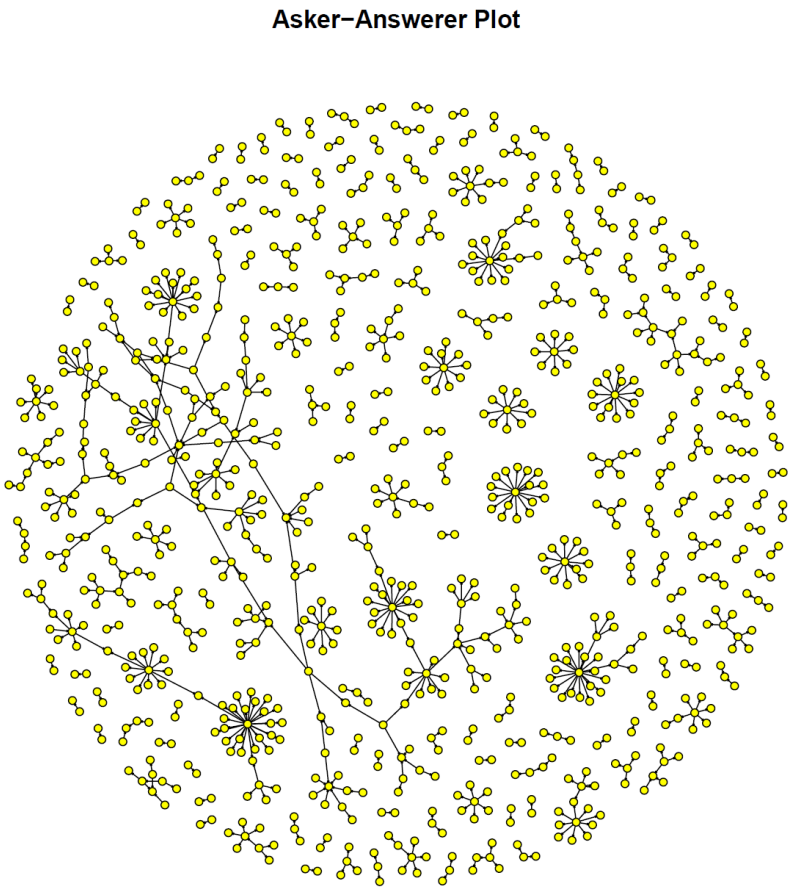
```
#library to plot
library(igraph)
#data frame is assigned to network with the directed links
network <- graph_from_data_frame(d=link, directed=T)
```

Step 3:

- Plotting the graph for the above network with layout 'fruchterman.reingold' with some specification for better visualization.

```
#Plotting is done with 'fruchterman.reingold' layout
plot(network,vertex.size = 2,vertex.color = 'yellow',edge.width = 1,
     edge.color = 'black',edge.arrow.size=.02, vertex.label = NA,
     layout = layout.fruchterman.reingold,main = "Asker-Answerer Plot")
```

Output:



Step 4:

- Importing library 'CINNA' to find the giant component of the above network.

```
library(CINNA)
#Gaint link from above plotted network is assigned to com
com<-giant_component_extract(network,directed = TRUE)
```

Step 5:

- All the user id's of the giant component is data framed into giant as shown.

```
#user id's in the giant network is sent to giant
giant<-data.frame(
  asker=com[2][[1]][,1],
  answer=com[2][[1]][,2])
```

Step 6:

- Above giant is assigned to g_network with the directed graph.

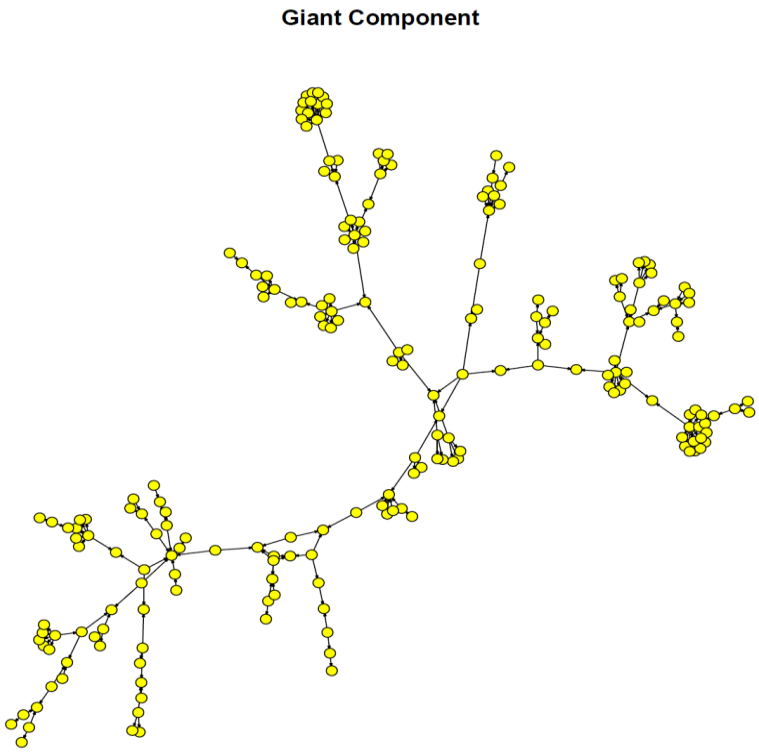
```
#data frame is assigned to g_network with the directed links
g_network<-graph_from_data_frame(d=giant, directed=T)
```

Step 7:

- G_network is plotted as plotted in the previous plot.

```
#Plotting is done with 'fruchterman.reingold' layout
plot(g_network,vertex.size = 3,vertex.color = 'yellow',edge.width = 1,
     edge.color = 'black',edge.arrow.size=.04, vertex.label = NA,
     layout = layout.fruchterman.reingold,main = "Giant Component")
```

Output:



Step 8:

- Output file giant_component.tsv is generated for the data in ‘giant’.

```
#Output file is generated in .tsv
write.table(giant, file='giant_component123.tsv', quote=FALSE, sep='\t', col.names = NA)
```

Output file:

Problems faced:

- As I have mentioned in the first stage throttling has been the first issue as we are fetching data from a website, I have overcome by researching about StackAPI and found a solution to create an application and use the secret key while using API.
- The next issue is selecting the appropriate tag and filters to get enough number of links, I have used Stack Exchange database to select my tag. Initially, I have used Java and sorted by votes, where I got nearly 100-200 answers for each question. Which gives me a huge data.
- So, later I have selected tag ‘python’ and sorted by ‘activity’ which gave me 700-800.
- As R-code is new to me, Initially it took some time to plot the giant component but later by using some library and commands it made easy to find the giant component.

GitHub: https://github.com/AkhilGonna/StackOverflow_Q-A

G-Drive: https://drive.google.com/drive/u/1/folders/0AMlBhbbA_ogPUk9PVA