ECE 720: Assignment #1

Igor Malis (1102098)
imalis@ualberta.ca
University of Alberta

Abstract

This assignment involves an exploration of Stack Overflow, including it's data model and application programming interface (API). After picking a set of tags (used as a filter to retrieve posts), the Stack Overflow API will be used to retrieve data relating to those posts, which will be saved in TSV. After retrieving the data, it will be imported into R, where the igraph package will be used to plot the graph, and the CINNA package will be used to extract its giant component.

1 Introduction

Stack Overflow (SO), located at https://stackoverflow.com/, is a social network for Software Developers/Engineers, which allows users to post questions and answers for technical, programming-related topics. It allows users to attach tags to posts (such as the programming language or technology the question is about), and allows users to up-vote or down-vote questions and answers. Every registered user on the platform has a reputation rating and collection of badges they have received, making a person's previous experience and level of expertise transparent and visible to all viewers. As part of this assignment, I will explore Stack Overflow, and utilize its API to retrieve post data and use it to plot a network graph with R.

2 Picking Tags

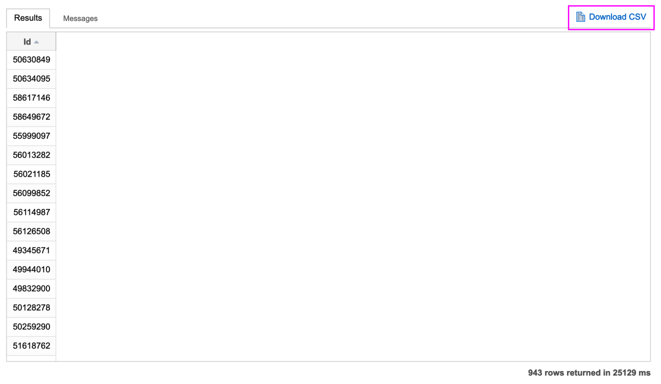
Prior to beginning the data analysis, it was first necessary to determine a series of tags which would result in 500 - 1000 Stack Overflow posts being returned. In order to do this, the Stack Exchange data explorer was utilized. By trial and error, the SQL query shown in Figure 1 was obtained (which returns 943 posts).

This results in four tags used together:

- tensorflow
- keras
- machine-learning

```
SELECT Id
FROM Posts
WHERE Tags LIKE '%tensorflow%'
  AND Tags LIKE '%keras%'
  AND Tags LIKE '%machine-learning%'
  AND Tags LIKE '%python%'
```

Figure 1. SQL query used on Stack Exchange Data Explorer



Id
50630849
50634095
58617146
58649672
55999097
56013282
56021185
56099852
56114987
56126508
49345671
49944010
49832900
50128278
50259290
51618762

943 rows returned in 25129 ms

Figure 2. Post ID's returned by Stack Exchange Data Explorer

- python

After picking the tags, the data explorer provided a listing of all question ID's associated with the SQL query in Figure 1. The 'Download CSV' feature was used to download the ID's of all posts related to the tags (see Figure 2).

3 Retrieving Data via StackOverflow API

The Stack Overflow API (https://api.stackexchange.com/docs) was used to retrieve data about each post that was included as part of the query above. The following information was obtained by calling the Stack Overflow API:

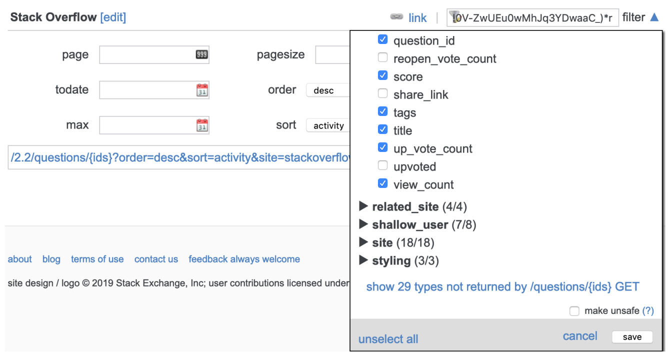
- Question/answer ID
- User ID (or display_name, if the user is unregistered)
- Post title, body (markdown) and tags
- The following dates: created, last_activity
- Other fields related to posts, such as is_answered, is_accepted, link, score, view_count, up_vote_count, and down_vote_count

In order for the API to return all of the above fields within a single API call, the filter query string parameter had to be specified. The value of the filter was obtained by using Stack Overflow API documentation (see https://api.stackexchange.com/docs/questions-by-ids). The resulting filter string that was obtained by including all fields as required above is: `!0V-ZwUEu0wMhJq3YDwaaC_)*r` (see Figure 3).

A python script was created to retrieve the above information from the API, and save it to a file (see Appendix). Four files were generated by this script (these are located in the /data/ directory):

- *allPosts.tsv*: contains all fields mentioned above
- *allPosts-metaData.tsv*: contains only question/answer ID, post type (question or answer), and user ID
- *askerAnswerer_nodes.tsv*: contains only the node IDs for asker-answerer graph (the username or display_name)
- *askerAnswerer_edges.tsv*: contains edges for asker-answerer graph

Note: the graph "askerAnswerer.tsv" was exported as two separate files, one containing nodes, and the other containing edges (instead of a single file), to ensure that questions with no answers appeared on the resulting graph.



Stack Overflow [edit]

link | !0V-ZwUEu0wMhJq3YDwaaC_)*r filter

page	
todate	
max	
pagesize	
order	desc
sort	activity

/2.2/questions/{ids}?order=desc&sort=activity&site=stackoverflo

- ✓ question_id
- ☐ reopen_vote_count
- ✓ score
- ☐ share_link
- ✓ tags
- ✓ title
- ✓ up_vote_count
- ☐ upvoted
- ✓ view_count

related_site (4/4)
shallow_user (7/8)
site (18/18)
styling (3/3)

show 29 types not returned by /questions/{ids} GET

☐ make unsafe (?)

about blog terms of use contact us feedback always welcome

site design / logo © 2019 Stack Exchange, Inc; user contributions licensed under

Figure 3. Selected filter on Stack Exchange Data Explorer

4 Plotting Network Graph

The data exported in the previous section was then imported into R. A network graph was generated from this data using the igraph library. Due to the graph being very cluttered, the following settings were used when drawing the resulting graph:

- vertex_size = 1
- vertex.label = NA
- edge.width = 0.25
- edge.arrow.size = 0.1
- edge.arrow.width = 0.5
- layout *component_layouts* was used to improve visibility of the data

The resulting graph is included with this submission as file *askerAnswerer.pdf*, and is shown in Figure 4 (the edges are hard to see in this picture; please see PDF included with submission).

5 Exporting Giant Graph

Lastly, the giant component of the graph was extracted from the original graph using R's CINNA package. This graph is also included as part of the submission: *askerAnswerer-giant.pdf*, and is shown in Figure 5.

6 Conclusions

Stack Overflow is a very popular online software social network. In this assignment, I had the chance to explore the Stack Overflow API, retrieve a small data set with it using Python, and plot the resulting network graphs using R's igraph and CINNA libraries.

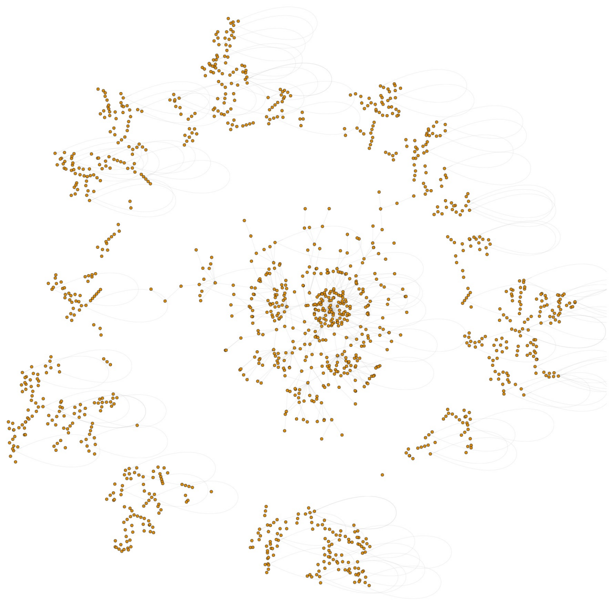


Figure 4. Entire network graph

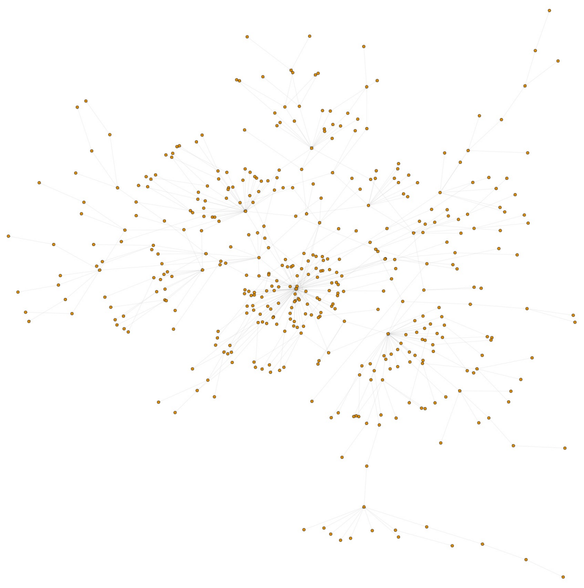


Figure 5. Giant component of network graph

A code.py

```
import requests
from pandas import read_csv
import numpy as np
import time

# Read in Post ID’s retrieved from Stack Exchange Data Explorer
question_ids = read_csv(’./data/QueryResults.csv’).values[:,0]

# Split into 10 chunks with < 100 questions per array
split_num = np.array_split(question_ids, 10)
split_str = []

for i in split_num:
    split_str.append( ’;’.join( str(x) for x in i ) )

# Write file headers
with open(’./data/allPosts.tsv’, ’w+’) as output:
    output.write(’question_id\tanswer_id\tuser_id\tpost_type\tlink\tscore\tcreated\tlast_activity\ttitle\
        tbody_markdown\tis_answered\tis_accepted\tanswer_count\tview_count\tup_vote_count\
        tdown_vote_count\ttags\n’)
with open(’./data/allPosts-metaData.tsv’, ’w+’) as output:
    output.write(’question_id\tanswer_id\tuser_id\tpost_type\n’)
with open(’./data/askerAnswerer_nodes.tsv’, ’w+’) as output:
    output.write(’id\n’)
with open(’./data/askerAnswerer_edges.tsv’, ’w+’) as output:
    output.write(’from\tto\tweight\ttype\n’)

users = []

# Save question data to tsv files
for chunk in split_str:
    url = ’https://api.stackexchange.com/2.2/questions/{}?pagesize=100&site=stackoverflow&filter=!0V-
        ZwUEu0wMhJq3YDwaaC_)*r’.format(chunk)
    r = requests.get(url = url)
    data = r.json()
    for item in data[’items’]:
        with open(’./data/allPosts.tsv’, ’a+’) as output:
            output.write(’{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\n’.format(
                item[’question_id’],
                ’’,
                item[’owner’][’user_id’] if item[’owner’][’user_type’] == ’registered’ else item[’owner
                    ’][’display_name’],
                ’question’,
                item[’link’],
                item[’score’],
                time.ctime(item[’creation_date’]),
                time.ctime(item[’last_activity_date’]),
                item[’title’],
                item[’body_markdown’].replace(’\r\n’, ’\\n’).replace(’\t’, ’\\t’),
                item[’is_answered’],
                ’’,
                item[’answer_count’],
                item[’view_count’],
                item[’up_vote_count’],
                item[’down_vote_count’],
        ';'.join(item['tags'])
    ))
with open('./data/allPosts-metaData.tsv', 'a+') as output:
    output.write('{}\t{}\t{}\t{}\n'.format(
        item['question_id'],
        '',
        item['owner']['user_id'] if item['owner']['user_type'] == 'registered' else item['owner
            ']['display_name'],
        'question'
    ))

users.append(item['owner']['user_id'] if item['owner']['user_type'] == 'registered' else item['
    owner']['display_name'])

if 'answers' in item:
    for answer in item['answers']:
        with open('./data/allPosts.tsv', 'a+') as output:
            output.write('{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\t{}\n'.
                format(
                answer['question_id'],
                answer['answer_id'],
                answer['owner']['user_id'] if answer['owner']['user_type'] == 'registered' else
                    answer['owner']['display_name'],
                'answer',
                '',
                answer['score'],
                time.ctime(answer['creation_date']),
                time.ctime(answer['last_activity_date']),
                '',
                answer['body_markdown'].replace('\r\n', '\\n').replace('\t', '\\t'),
                '',
                answer['is_accepted'],
                '',
                '',
                answer['up_vote_count'],
                answer['down_vote_count'],
                ';'.join(item['tags'])
            ))
        with open('./data/allPosts-metaData.tsv', 'a+') as output:
            output.write('{}\t{}\t{}\t{}\n'.format(
                answer['question_id'],
                answer['answer_id'],
                answer['owner']['user_id'] if answer['owner']['user_type'] == 'registered' else
                    answer['owner']['display_name'],
                'answer'
            ))

        with open('./data/askerAnswerer_edges.tsv', 'a+') as output:
            output.write('{}\t{}\t{}\t{}\n'.format(
                item['owner']['user_id'] if item['owner']['user_type'] == 'registered' else item['
                    owner']['display_name'],
                answer['owner']['user_id'] if answer['owner']['user_type'] == 'registered' else
                    answer['owner']['display_name'],
                '1',
                'askerAnswerer'
            ))
            users.append(answer['owner']['user_id'] if answer['owner']['user_type'] == 'registered'
                else answer['owner']['display_name'])

# Remove duplicate users
users = list(set(users))

with open('./data/askerAnswerer_nodes.tsv', 'a+') as output:
    for user in users:
        output.write('{}\n'.format(user))
```

B code.r

```
library(igraph)
library(CINNA)

# Read CSV files exported by code.py
nodes <- read.csv("Documents/ECE720/asn1/data/askerAnswerer_nodes.tsv", sep="\t")
edges <- read.csv("Documents/ECE720/asn1/data/askerAnswerer_edges.tsv", sep="\t")

# Generate graph
net <-graph.data.frame(edges, nodes, directed=T)

# Save graph to file askerAnswerer.pdf
pdf(file="Documents/ECE720/asn1/askerAnswerer.pdf")
plot(net, vertex.size=1, vertex.label=NA, edge.width=0.25, edge.arrow.size=.1, edge.arrow.width=0.5,
    layout=layout_components(net))
dev.off()

# Save graph to file askerAnswerer.jpg
jpeg(file="Documents/ECE720/asn1/askerAnswerer.jpg", width = 480*4, height = 480*4)
plot(net, vertex.size=1, vertex.label=NA, edge.width=0.25, edge.arrow.size=.1, edge.arrow.width=0.5,
    layout=layout_components(net))
dev.off()

# Get giant component of graph and save as PDF
net2 = giant_component_extract(net)
pdf(file="Documents/ECE720/asn1/askerAnswerer-giant.pdf")
plot(net2[[1]], vertex.size=1, vertex.label=NA, edge.width=0.25, edge.arrow.size=.1, edge.arrow.width
    =0.5, layout=layout_components(net2[[1]]))
dev.off()

# Save giant component as JPG
jpeg(file="Documents/ECE720/asn1/askerAnswerer-giant.jpg", width = 480*4, height = 480*4)
plot(net2[[1]], vertex.size=1, vertex.label=NA, edge.width=0.25, edge.arrow.size=.1, edge.arrow.width
    =0.5, layout=layout_components(net2[[1]]))
dev.off()

# Write giant component of graph to new TSV file
write.table(net2[[2]], file="Documents/ECE720/asn1/data/asker-answerer-giant.tsv", quote=FALSE, sep="\t",
     col.names=NA)
```