

```

1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import datetime
4 import numpy as np
5
6 import twitter_getting_data as tgd
7
8 def parse_net_report(path):
9     file = open(path)
10    relations = eval(file.readline())
11    file.close()
12    ids = [] # list of ids involved
13    for relationship in relations:
14        if not (relationship['source_user_id'] in ids):
15            ids.append(relationship['source_user_id'])
16
17    edges = {'followship': [],
18            'likes': [],
19            'rts': [],
20            'replies': [],
21            'matched_likes': [],
22            'matched_rts': []}
23
24    for relation in relations:
25        if relation['source_follows_target']:
26            edges['followship'].append((relation['source_user_id'],
27                                       relation['target_user_id'],
28                                       1))
29
30        if relation['favorites']:
31            edges['likes'].append((relation['source_user_id'],
32                                  relation['target_user_id'],
33                                  len(relation['favorites'])))
34
35        if relation['retweeted']:
36            edges['rts'].append((relation['source_user_id'],
37                                relation['target_user_id'],
38                                len(relation['retweeted'])))
39
40        if relation['replies']:
41            edges['replies'].append((relation['source_user_id'],
42                                     relation['target_user_id'],
43                                     len(relation['replies'])))
44
45        if (relation['matched_likes'] and \
46            relation['source_user_id'] < relation['target_user_id']):
47            # we only add matched_likes once
48            edges['matched_likes'].append((relation['source_user_id'],
49                                           relation['target_user_id'],
50                                           len(relation['matched_likes'])))
51
52        if (relation['matched_rts'] and \
53            relation['source_user_id'] < relation['target_user_id']):
54            # we only add matched_rts once
55            edges['matched_rts'].append((relation['source_user_id'],
56                                         relation['target_user_id'],
57                                         len(relation['matched_rts'])))
58
59    return ids, edges
60
61 def get_directed_overall_edges(ids, edges_dict):
62     dedges = []
63     for source_id in ids:
64         for target_id in ids:
65             if source_id != target_id:
66                 weight = 0
67                 for pairs in edges_dict['followship']:
68                     if source_id == pairs[0] and \
69                        target_id == pairs[1]:
70                         weight += pairs[2]
71                 for pairs in edges_dict['likes']:
72                     if source_id == pairs[0] and \
73                        target_id == pairs[1]:
74                         weight += pairs[2]
75                 for pairs in edges_dict['rts']:
76                     if source_id == pairs[0] and \
77                        target_id == pairs[1]:

```

```

77         weight += pairs[2]
78     for pairs in edges_dict['replies']:
79         if source_id == pairs[0] and \
80             target_id == pairs[1]:
81             weight += pairs[2]
82     if weight != 0:
83         dedges.append((source_id, target_id, weight))
84     return dedges
85
86
87 def get_user_stats(likes, timeline):
88     """
89     time in days.
90     """
91
92     stats = {'like_time_period': datetime.datetime.strptime(likes[0]['created_at'],
93     "%a %b %d %H:%M:%S %z %Y") \
94         - datetime.datetime.strptime(likes[-1]['created_at'
95     ], "%a %b %d %H:%M:%S %z %Y"),
96     'timeline_time_period': datetime.datetime.strptime(timeline[0]['
97     created_at'], "%a %b %d %H:%M:%S %z %Y") \
98         - datetime.datetime.strptime(timeline[-1]['
99     created_at'], "%a %b %d %H:%M:%S %z %Y"),
100     'total_likes': len(likes),
101     'total_timeline': len(timeline),
102     'likes_ranking': {},
103     'rts_ranking': {},
104     'replies_ranking': {}
105     }
106
107     for like in likes:
108         stats['likes_ranking'][like['user']['id']] = 1 + stats['likes_ranking'].get(
109     like['user']['id'], 0)
110
111     for tweet in timeline:
112         if tweet['in_reply_to_user_id']:
113             stats['replies_ranking'][tweet['in_reply_to_user_id']] = 1 + stats['
114     replies_ranking'].get(tweet['in_reply_to_user_id'], 0)
115
116         if tweet.get('retweeted_status', False):
117             stats['rts_ranking'][tweet['retweeted_status']['user']['id']] = 1 +
118     stats['rts_ranking'].get(tweet['retweeted_status']['user']['id'], 0)
119
120     return stats
121
122
123 def get_eigcentrality(potential_groupmates, net):
124     id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}
125     N = len(potential_groupmates)
126     adjacency = np.zeros((N, N))
127     for i, j in net:
128         if i != j:
129             adjacency[id_to_index[i], id_to_index[j]] = 1
130             adjacency[id_to_index[j], id_to_index[i]] = 1
131     vals, vecs = np.linalg.eigh(adjacency)
132     index = np.where(vals == max(vals))[0]
133     centrality = abs(np.transpose(vecs[:, index])[0])
134     return centrality
135
136
137 def get_group_involvement(initial_friends, potential_groupmates, net,
138     max_iteration=100, partial_increment=0.1, decrease_factor=
139     1.05):
140     id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}
141     N = len(potential_groupmates)
142     adjacency = np.zeros((N, N))
143     for i, j in net:
144         if i != j:
145             adjacency[id_to_index[i], id_to_index[j]] = 1
146             adjacency[id_to_index[j], id_to_index[i]] = 1
147
148     node_rank = np.zeros(len(potential_groupmates))
149     for uid in initial_friends:
150         if uid in potential_groupmates:
151             node_rank[id_to_index[uid]] = 1
152
153     iter = 0

```

```

145     while iter < max_iteration:
146         new_rank = np.zeros(len(potential_groupmates))
147         for uid in potential_groupmates:
148             score = node_rank[id_to_index[uid]]
149             if sum(adjacency[id_to_index[uid], :]) > 0: # this should always happen
150                 new_rank += score / decrease_factor \
151                     * adjacency[id_to_index[uid], :] / sum(adjacency[
id_to_index[uid], :])
152         node_rank = new_rank
153         for uid in initial_friends:
154             if uid in potential_groupmates:
155                 node_rank[id_to_index[uid]] += partial_increment
156         iter += 1
157     return node_rank
158
159
160 def get_like_involvement(initial_friends, potential_groupmates, net, mlikes=False,
161                          max_iteration=100, partial_increment=0.1, decrease_factor=
1.05):
162     id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}
163     N = len(potential_groupmates)
164     adjacency = np.zeros((N, N))
165     for i, j, likeij, likeji, mlikes in net:
166         if i != j:
167             if not mlikes:
168                 adjacency[id_to_index[i], id_to_index[j]] = likeij
169                 adjacency[id_to_index[j], id_to_index[i]] = likeji
170             else:
171                 adjacency[id_to_index[i], id_to_index[j]] = mlikes
172                 adjacency[id_to_index[j], id_to_index[i]] = mlikes
173
174     node_rank = np.zeros(len(potential_groupmates))
175     for uid in initial_friends:
176         if uid in potential_groupmates:
177             node_rank[id_to_index[uid]] = 1
178     iter = 0
179     while iter < max_iteration:
180         new_rank = np.zeros(len(potential_groupmates))
181         for uid in potential_groupmates:
182             score = node_rank[id_to_index[uid]]
183             if sum(adjacency[id_to_index[uid], :]) > 0: # this should always happen
184                 new_rank += score / decrease_factor \
185                     * adjacency[id_to_index[uid], :] / sum(adjacency[
id_to_index[uid], :])
186         node_rank = new_rank
187         for uid in initial_friends:
188             if uid in potential_groupmates:
189                 node_rank[id_to_index[uid]] += partial_increment
190         iter += 1
191     return node_rank
192
193
194 def get_like_edges(ids, path):
195     net = [] # (usr1, usr2, llikes2, 2likes1, matched_likes)
196     with open(path) as liked_file:
197         liked = {}
198         for line in liked_file:
199             key, value = line.split(':', maxsplit=1)
200             liked[int(key)] = eval(value)
201
202     index = 0
203     for source_id in ids:
204         index += 1
205         for target_id in ids[index:]:
206             likes12 = 0
207             likes21 = 0
208             mlikes = 0
209
210             for tweet in liked[source_id]:
211                 if target_id == tweet['user']['id']:
212                     likes12 += 1
213                 #if tweet['id'] in liked[target_id]: # Don't think this is working
214                 #    mlikes += 1
215             for tweet in liked[target_id]:
216                 if source_id == tweet['user']['id']:
217                     likes21 += 1

```

```

file - network_analysis.py
218
219         if likes12+likes21+mlikes > 0:
220             net.append((source_id, target_id, likes12, likes21, mlikes))
221     return net
222
223
224 def expanding_the_net():
225     test_names = *** # Python list of length 5 which contains the Twitter user names
, as strings, of each member in the seed-group
226     test_ids = *** # Python list of length 5 which contains the Twitter user IDs, as
integers, of each member in the seed-group
227
228     file = open('large_net/qmisanz_friends_ids')
229     potential_groupmates = eval(file.readline())
230     file.close()
231     file = open('large_net/qmisanz_friends_ids_netdata')
232     net = []
233     for pair in file:
234         net.append(eval(pair))
235     file.close()
236
237     #centrality = get_eigcentrality(potential_groupmates, net)
238     #node_rank = get_group_involvement(test_ids, potential_groupmates, net)
239     like_net = get_like_edges(potential_groupmates, 'large_net/
qmisanz_freinds_liked_2020-04-19')
240     node_rank = get_like_involvement(test_ids, potential_groupmates, like_net)
241
242     pairs = [(nr, uid) for nr, uid in zip(node_rank, potential_groupmates)]
243     pairs.sort(reverse=True)
244     node_rank = np.array([nr for nr, uid in pairs])
245     potential_groupmates = np.array([uid for nr, uid in pairs])
246     id_to_index = {uid: index for index, uid in enumerate(potential_groupmates)}
247
248     show_plot = False
249     if show_plot:
250         for nr, uid in pairs:
251             if uid in test_ids:
252                 plt.plot(id_to_index[uid], nr, '.C1')
253             else:
254                 plt.plot(id_to_index[uid], nr, '.C0')
255         plt.show()
256
257     candidate_ranking = open('large_net/candidate_ranking_directed_likes', 'w')
258
259     idtoname_file = open('large_net/qmisanz_friends_idtouser', 'r')
260     id_to_name = eval(idtoname_file.readline())
261     idtoname_file.close()
262
263     for nr, uid in pairs:
264         if uid in test_ids:
265             print('!! {:17}, {:20}, {:8.6f}'.format(id_to_name[uid], uid, nr), file=
candidate_ranking)
266         else:
267             print('{:20}, {:20}, {:8.6f}'.format(id_to_name[uid], uid, nr), file=
candidate_ranking)
268     candidate_ranking.close()
269
270
271 if __name__ == '__main__':
272     test_names = *** # Python list of length 10 which contains the Twitter user
names, as strings, of each member in the test group
273     test_ids = *** # Python list of length 10 which contains the Twitter user IDs,
as integers, of each member in the test group
274
275     idtoname = {uid:name for uid, name in zip(test_ids, test_names)}
276     idtonumber = {uid:number for number, uid in enumerate(test_ids)}
277
278     # The time stamp in the name of the files should be adjusted, this is just an
example
279     ids, edges = parse_net_report('twitter_out/net_interactions_2020-06-08 16:29:58.
015124.data')
280
281     # printo name to number
282     file_nodenames = open('twitter_out/edges_report_2020-06-08/node_names_2020-06-08
.txt', 'w')
283     print('#COMMENT: Each line contains the twitter screen name and the' +

```

```

file /network_analysis.py
284     ' node number assigned to each user.', file=file_nodenames)
285     for number, name in enumerate(test_names):
286         print('{} {}'.format(name, number), file=file_nodenames)
287     file_nodenames.close()
288
289     # print followship relations
290     file_followship = open('twitter_out/edges_report_2020-06-08/
edges_followship_2020-06-08.txt', 'w')
291     print('#COMMENT: Each line contains a follow relation from'+
292           ' the first user to the second, each identified by their node number'+
293           ' If the third number is 0 there is no relation.', file=file_followship)
294     for src in ids:
295         for trgt in ids:
296             if src == trgt:
297                 continue
298             if (src, trgt, 1) in edges['followship']:
299                 print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 1), file=
file_followship)
300             else:
301                 print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=
file_followship)
302     file_followship.close()
303
304     # print like relations
305     file_like = open('twitter_out/edges_report_2020-06-08/edges_like_2020-06-08.txt'
, 'w')
306     print('#COMMENT: Each line contains the information about how many likes the
first user' +
307           ' gives to the second, each identified by their node number.', file=
file_like)
308     for src in ids:
309         for trgt in ids:
310             if src == trgt:
311                 continue
312             for contact in edges['likes']:
313                 if (src, trgt) == contact[:2]:
314                     print('{} {} {}'.format(idtonumber[src], idtonumber[trgt],
contact[2]), file=file_like)
315                     break
316             else:
317                 print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=
file_like)
318     file_like.close()
319
320     # print matched like relations
321     file_mlike = open('twitter_out/edges_report_2020-06-08/edges_matched_like_2020-
06-08.txt', 'w')
322     print('#COMMENT: Each line contains the information about how many times both
users'+
323           ' liked the same tweet, each user identified by their node number.', file=
file_mlike)
324     for src in ids:
325         for trgt in ids:
326             if src == trgt:
327                 continue
328             for contact in edges['matched_likes']:
329                 if (src, trgt) == contact[:2]:
330                     print('{} {} {}'.format(idtonumber[src], idtonumber[trgt],
contact[2]), file=file_mlike)
331                     break
332             else:
333                 print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=
file_mlike)
334     file_mlike.close()
335
336     # print rt relations
337     file_rt = open('twitter_out/edges_report_2020-06-08/edges_rt_2020-06-08.txt', 'w
')
338     print('#COMMENT: Each line contains the information about how many times the
first user' +
339           ' retweeted the second one, each identified by their node number.', file=
file_rt)
340     for src in ids:
341         for trgt in ids:
342             if src == trgt:
343                 continue

```

```

file /network_analysis.py
344         for contact in edges['rts']:
345             if (src, trgt) == contact[:2]:
346                 print('{} {} {}'.format(idtonumber[src], idtonumber[trgt],
contact[2]), file=file_rt)
347                 break
348             else:
349                 print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=
file_rt)
350         file_rt.close()
351
352         # print reply relations
353         file_reply = open('twitter_out/edges_report_2020-06-08/edges_reply_2020-06-08.
txt', 'w')
354         print('#COMMENT: Each line contains the information about how many times the
first user' +
355             ' replied the second, each identified by their node number.', file=
file_reply)
356         for src in ids:
357             for trgt in ids:
358                 if src == trgt:
359                     continue
360                 for contact in edges['replies']:
361                     if (src, trgt) == contact[:2]:
362                         print('{} {} {}'.format(idtonumber[src], idtonumber[trgt],
contact[2]), file=file_reply)
363                         break
364                 else:
365                     print('{} {} {}'.format(idtonumber[src], idtonumber[trgt], 0), file=
file_reply)
366         file_reply.close()

```