```
Help
                                                                                                                                         Python 3 O
                                             Widgets
 File
       Edit
              View
                              Cell
                                     Kernel
                      Insert
                                                                                                                           Not Trusted
                             ► Run ■ C → Code
↑ ↓
                                                            ~
       In [1]:
                 1 # Delicious
                 2 file1 = open(r'C:\Users\DELL\Downloads\Compressed\delicious_UsrResTag\delicious_UsrResTag', 'r', encoding="utf8")
                 3 Lines = file1.readlines()
                 4 print(len(Lines))
                140126555
               Preprocessing
       In [3]:
                 1 from tqdm import tqdm
                 2 node1_List = []
                 3 node2_List = []
                 4 time_List = []
                 5 # Strips the newline character
                 6 for line in tqdm(range(0, 1000000)):
                        time, node1, node2, _ = Lines[line].split("\t")
                 8
                        node1 List.append(node1)
                        node2_List.append(node2)
                 9
                        time_List.append(time)
                10
                                                                                 | 1000000/1000000 [00:01<00:00, 923182.64it/s]
       In [4]:
                 1 import networkx as nx
                 2 G = nx.Graph()
       In [5]:
                 1 for edge in tqdm(range(len(node1_List))):
                    node1=node1_List[edge]
                     node2=node2_List[edge]
                       G.add_edge(node1,node2)
                 5 Degree_count_from={}
                 6 Degree_count_to={}
                 7 Degree_from={}
                 8 Degree_to={}
                 9 node1_List_set=set(node1_List)
                10 node2_List_set=set(node2_List)
                11
                12 | for x in node1_List_set:
                # if(x not in Degree count from):
                          Degree_from[x]=G.degree(x)
                14
                     # else:
                15
                     # print("hi")
                16
                      # break
                18
                19 for y in node2_List_set:
                     # if(x not in Degree_count_to):
                21
                          Degree_to[y]=G.degree(y)
                22
                                            1000000/1000000 [15:59<00:00, 1042.47it/s]
               Degree Distribution : Users
                 1 list_of_degrees=list(Degree_from.values())
       In [6]:
                 2 from_degree_counts={}
                 3 for deg in list_of_degrees:
                        if(deg in from_degree_counts):
                            from_degree_counts[deg]+=1
                 5
                 6
                        else:
                 7
                            from_degree_counts[deg]=1
                 8 max_from_degree_freq=max(from_degree_counts.values())
                10
                11 | from_degree_counts1=from_degree_counts.copy()
                12
                13 from_distis=dict((sorted(from_degree_counts1.items())))
                14 from_max_degree_count=max(from_distis.values())
                15 # total_nodes=sum(distis.values())
                16 # from_distis.update((x, round(y/from_max_degree_count,2)) for x, y in from_distis.items())
                17 # x_axis=list(from_distis.keys())[1:len(from_distis.keys())]
                18 # y_axis=list(from_distis.values())[1:len(from_distis.values())]
                20 x_axis=list(from_distis.keys())
                21 y_axis=list(from_distis.values())
                22
                23 import matplotlib.pyplot as plt
                24 import matplotlib
                25 plt.loglog(x_axis, y_axis, "o")
                26 # plt.xscale("log")
                27 # plt.yscale("log")
                28 plt.xlabel('degree')
                30 plt.ylabel('number of nodes having degree')
                32 plt.title('Degree Distribution of Users')
                33 plt.legend()
                34 # plt.show()
                35 | fig = matplotlib.pyplot.gcf()
                No handles with labels found to put in legend.
                                               Degree Distribution of Users
                   10^{3}
                number of nodes having degree
                   10°
                        10°
                                          10¹
                                                           10^{2}
                                                                             10^{3}
                                                        degree
               Degree Distribution : Tags
                 1 list_of_degrees=list(Degree_to.values())
       In [7]:
                 2 to_degree_counts={}
                 3 for deg in list_of_degrees:
                        if(deg in to_degree_counts):
                            to_degree_counts[deg]+=1
                 6
                        else:
                            to_degree_counts[deg]=1
                   max_to_degree_freq=max(to_degree_counts.values())
                10
                11 to_degree_counts1=to_degree_counts.copy()
                12
                13 to_distis=dict((sorted(to_degree_counts1.items())))
                14 | to_max_degree_count=max(to_distis.values())
                15 # total_nodes=sum(distis.values())
                # to_distis.update((x, round(y/to_max_degree_count,2)) for x, y in to_distis.items())
                17 # x_axis=list(to_distis.keys())[1:len(to_distis.keys())]
                18 # y_axis=list(to_distis.values())[1:len(to_distis.values())]
                19
                20 x_axis1=list(to_distis.keys())
                21 y_axis1=list(to_distis.values())
                22
                23 import matplotlib.pyplot as plt
                24 import matplotlib
                25 # plt.scatter(x_axis1, y_axis1, label = "fraction")
                26 plt.loglog(x_axis1, y_axis1,"o")
                27 # plt.xscale("log")
                28 # plt.yscale("log")
                29 plt.xlabel('degree')
                30
                31 plt.ylabel('number of nodes having degree')
                33 plt.title('Degree Distribution of Tags')
                34 plt.legend()
                35 # plt.show()
                36 fig = matplotlib.pyplot.gcf()
               No handles with labels found to put in legend.
                                               Degree Distribution of Tags
                   10<sup>5</sup>
                number of nodes having degree
                   10^{1}
                   10°
                                                              10^{2}
                                           10<sup>1</sup>
                                                                                 10^{3}
                        10°
                                                        degree
      In [11]:
                 1 node1_degree_dict = {}
                 2 node2_degree_dict = {}
                 3 total_degree = 0
                 4 node_time_dict = {}
                 5 for i in tqdm(range(0, len(node2_List))):
                        node1 = node1 List[i]
                        node2 = node2_List[i]
                 8
                        if node1 not in node1_degree_dict:
                            node1 degree dict[node1] = 0
                 9
                10
                        node1_degree_dict[node1] += 1
                11
                        if node2 not in node2 degree dict:
                            node2_degree_dict[node2] = 0
                13
                14
                        node2 degree dict[node2] += 1
                15
                        total_degree += 2
                16
                                                                                 | 1000000/1000000 [00:01<00:00, 941337.64it/s]
                 1 probability_dict = {}
      In [12]:
                 for key, value in node1_degree_dict.items():
                        probability_dict[key] = value/total_degree
               Network Evolution : Probability vs Degree
      In [13]:
                 1 import matplotlib.pyplot as plt
                 2 import matplotlib
                 4 plt.loglog(list(probability_dict.values()), list(node1_degree_dict.values()), "o")
                 5 # plt.scatter(list(probability_dict.values()), list(movie_degree_dict.values()), label = "scaled frequencies")
                 6 # plt.xscale("log")
                 7 # plt.yscale("log")
                 8 plt.xlabel('probability')
                10 plt.ylabel('degree')
                11
                12 plt.title('Evolution')
                13 plt.legend()
                14 # plt.show()
                15 fig = matplotlib.pyplot.gcf()
                16 fig.set_size_inches(10, 7)
               No handles with labels found to put in legend.
                                                       Evolution
                   10^{4}
                   10^{3}
                 eg 10<sup>2</sup>
                   10<sup>1</sup>
                   10°
                            10-6
                                           10-5
                                                          10^{-4}
                                                                          10^{-3}
                                                                                         10^{-2}
```

probability

Jupyter 42_Code3 Last Checkpoint: 7 minutes ago (autosaved)

Logout