ДОДАТОК А

Вихідний код програмних модулів

Neural_network_graph.ipynb

```
import networkx as nx
#Neural network graph
neuralGraph=nx.read_gml('celegansneural.gml')
nx.write_gexf(neuralGraph, "fileNeural.gexf", version="1.2draft")
#Degree centrality for neural network graph
degree_cent_neural = nx.degree_centrality(neuralGraph)
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
seventh_group=[];
eigth_group=[];
nineth_group=[];
ten_group=[];
for key,value in degree_cent_neural.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value < 0.1 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 255, 'a': 0}, 'size':30}
```

```
        first_group.append(value)
    if value > 0.1 and value < 0.2 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':40}
        second_group.append(value)
    if value > 0.2 and value < 0.3 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':50}
        third_group.append(value)
    if value > 0.3 and value < 0.4 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':60}
        fourth_group.append(value)
    if value > 0.4 and value < 0.5 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':70}
        fifth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(neuralGraph.nodes()))
places_degree={}
place_degree=1
for i in sorted(degree_cent_neural.items(), key=lambda pair: pair[1]):
    places_degree[i[0]]=place_degree
    place_degree+=1
print(places_degree)
print(degree_cent_neural.items())
nx.write_gexf(neuralGraph, "neural_graph_degree.gexf", version="1.2draft")
#Closeness centrality for neural network graph
closeness_cent_neural = nx.closeness_centrality(neuralGraph)
maximum=0;
minimum=1;
```

```
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
seventh_group=[];
eigth_group=[];
nineth_group=[];
ten_group=[];
for key,value in closeness_cent_neural.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value < 0.1 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 255, 'a': 0}, 'size':20}
        first_group.append(value)
    if value > 0.1 and value < 0.2 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':30}
        second_group.append(value)
    if value > 0.2 and value < 0.3 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':35}
        third_group.append(value)
    if value > 0.5 and value < 0.6 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':60}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
```

```
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(sixth_group))
places_closeness={}
place_closeness=1
for i in sorted(closeness_cent_neural.items(), key=lambda pair: pair[1]):
    places_closeness[i[0]]=place_closeness
    place_closeness+=1
print(places_closeness)
print(closeness_cent_neural.items())
nx.write_gexf(neuralGraph, "neural_graph_closeness.gexf", version="1.2draft")
#Betweenness centrality for neural network graph
betweenness_cent_neural = nx.betweenness_centrality(neuralGraph)
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
seventh_group=[];
eigth_group=[];
nineth_group=[];
ten_group=[];
for key,value in betweenness_cent_neural.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
```

```
    if value < 0.01 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 255, 'a': 0}, 'size':40}
        first_group.append(value)
    if value > 0.01 and value < 0.02 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':50}
        second_group.append(value)
    if value > 0.02 and value < 0.03 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':60}
        third_group.append(value)
    if value > 0.03 and value < 0.09 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':70}
        fourth_group.append(value)
    if value > 0.09 and value < 0.11 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':80}
        fifth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(neuralGraph.nodes()))
places_betweenness={}
place_betweenness=1
for i in sorted(betweenness_cent_neural.items(), key=lambda pair: pair[1]):
    places_betweenness[i[0]]=place_betweenness
    place_betweenness+=1
print(betweenness_cent_neural.items())
nx.write_gexf(neuralGraph, "neural_graph_betweenness.gexf", version="1.2draft")
#Eigenvector centrality for neural network graph
eigenvector_cent_neural = nx.eigenvector_centrality(neuralGraph)
maximum=0;
```

```
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
seventh_group=[];
eigth_group=[];
nineth_group=[];
ten_group=[];
for key,value in eigenvector_cent_neural.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value < 0.01 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 255, 'a': 0}, 'size':20}
        first_group.append(value)
    if value > 0.01 and value < 0.04 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':30}
        second_group.append(value)
    if value > 0.04 and value < 0.07 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':40}
        third_group.append(value)
    if value > 0.07 and value < 0.1 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':50}
        fourth_group.append(value)
    if value > 0.1 and value < 0.5 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':60}
        fifth_group.append(value)
```

```
    if value > 0.5 and value < 0.6 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':90}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(sixth_group))
print(len(neuralGraph.nodes()))
places_eigenvector={}
place_eigenvector=1
for i in sorted(eigenvector_cent_neural.items(), key=lambda pair: pair[1]):
    places_eigenvector[i[0]]=place_eigenvector
    place_eigenvector+=1
print(eigenvector_cent_neural.items())
nx.write_gexf(neuralGraph, "neural_graph_eigenvector.gexf", version="1.2draft")
#Pagerank centrality for neural network graph
pagerank_cent_neural = nx.pagerank(neuralGraph)
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
seventh_group=[];
eigth_group=[];
```

```
nineth_group=[];
ten_group=[];
for key,value in pagerank_cent_neural.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value < 0.001 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 255, 'a': 0}, 'size':15}
        first_group.append(value)
    if value > 0.001 and value < 0.002 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':20}
        second_group.append(value)
    if value > 0.002 and value < 0.003 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':30}
        third_group.append(value)
    if value > 0.003 and value < 0.004 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':40}
        fourth_group.append(value)
    if value > 0.004 and value < 0.008 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':40}
        fifth_group.append(value)
    if value > 0.1 and value < 0.13 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':90}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
```

```
print(len(fifth_group))
print(len(sixth_group))
print(len(neuralGraph.nodes()))
places_pagerank={}
place_pagerank=1
for i in sorted(pagerank_cent_neural.items(), key=lambda pair: pair[1]):
    places_pagerank[i[0]]=place_pagerank
    place_pagerank+=1
print(pagerank_cent_neural.items())
nx.write_gexf(neuralGraph, "neural_graph_pagerank.gexf", version="1.2draft")
#Load centrality for neural network graph
load_cent_neural = nx.load_centrality(neuralGraph)
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
seventh_group=[];
eigth_group=[];
nineth_group=[];
ten_group=[];
for key,value in load_cent_neural.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value < 0.01 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 255, 'a': 0}, 'size':20}
```

```
        first_group.append(value)
    if value > 0.01 and value < 0.02 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':30}
        second_group.append(value)
    if value > 0.02 and value < 0.03 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':40}
        third_group.append(value)
    if value > 0.03 and value < 0.04 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':40}
        fourth_group.append(value)
    if value > 0.04 and value < 0.09 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':60}
        fifth_group.append(value)
    if value > 0.09 and value < 0.11 :
        neuralGraph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':100}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(sixth_group))
print(len(neuralGraph.nodes()))
places_load={}
place_load=1
for i in sorted(load_cent_neural.items(), key=lambda pair: pair[1]):
    places_load[i[0]]=place_load
    place_load+=1
print(load_cent_neural.items())
nx.write_gexf(neuralGraph, "neural_graph_load.gexf", version="1.2draft")
#Summary measure
```

```
sum_places={}
degree_koeff=0.05
closeness_koeff=0.15
betweenness_koeff=0.1
eigenvector_koeff=0.2
pagerank_koeff=0.3
load_koeff=0.2
for key,value in degree_cent_neural.items():
   sum_places[key]=(degree_koeff * places_degree[key]) + (closeness_koeff * plac-
es_closeness[key]) + (betweenness_koeff * places_betweenness[key]) + (eigenvector_koeff * plac-
es_eigenvector[key]) + (pagerank_koeff * places_pagerank[key]) + (load_koeff * places_load[key])
for i in sorted(sum_places.items(), key=lambda pair: pair[1]):
   print(i)
```

Email_graph.ipynb

```
import networkx as nx
import csv
import pandas as pd
email_graph = nx.read_edgelist('email.txt', nodetype=int,
  data=(('Weight',float),), create_using=nx.DiGraph())
nx.write_gexf(email_graph, "email_graph_full.gexf", version="1.2draft")
#Degree centrality for email graph
degree_cent_email = nx.degree_centrality(email_graph)
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
for key,value in degree_cent_email.items():
```

```
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value > 0.1 and value < 0.2 :
        email_graph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':20}
        second_group.append(value)
    if value > 0.2 and value < 0.3 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':30}
        third_group.append(value)
    if value > 0.3 and value < 0.4 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':50}
        fourth_group.append(value)
    if value > 0.4 and value < 0.5 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':70}
        fifth_group.append(value)
    if value > 0.5 and value < 0.6 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':80}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(sixth_group))
places_degree={}
place_degree=1
for i in sorted(degree_cent_email.items(), key=lambda pair: pair[1]):
    places_degree[i[0]]=place_degree
```

```
    place_degree+=1
print(places_degree)
nx.write_gexf(email_graph, "email_graph_degree.gexf", version="1.2draft")
#Closeness centrality for email graph
closeness_cent_email = nx.closeness_centrality(email_graph)
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
for key,value in closeness_cent_email.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value > 0.1 and value < 0.2 :
        email_graph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':20}
        second_group.append(value)
    if value > 0.2 and value < 0.3 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':30}
        third_group.append(value)
    if value > 0.3 and value < 0.4 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':20}
        fourth_group.append(value)
    if value > 0.4 and value < 0.5 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':50}
        fifth_group.append(value)
    if value > 0.5 and value < 0.6 :
```

```
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':80}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(sixth_group))
places_closeness={}
place_closeness=1
for i in sorted(closeness_cent_email.items(), key=lambda pair: pair[1]):
    places_closeness[i[0]]=place_closeness
    place_closeness+=1
print(closeness_cent_email.items())
nx.write_gexf(email_graph, "email_graph_closeness.gexf", version="1.2draft")
#Betweenness centrality for email graph
betweenness_cent_email = nx.betweenness_centrality(email_graph)
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
for key,value in betweenness_cent_email.items():
    if value>maximum:
        key_max=key
        maximum=value
```

```
    if value<minimum:
        key_min=key
        minimum=value
    if value > 0.001 and value < 0.01 :
        email_graph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':30}
        second_group.append(value)
    if value > 0.01 and value < 0.03 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':40}
        third_group.append(value)
    if value > 0.03 and value < 0.05 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':50}
        fourth_group.append(value)
    if value > 0.07 and value < 0.1 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':60}
        fifth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
places_betweenness={}
place_betweenness=1
for i in sorted(betweenness_cent_email.items(), key=lambda pair: pair[1]):
    places_betweenness[i[0]]=place_betweenness
    place_betweenness+=1
print(betweenness_cent_email.items())
nx.write_gexf(email_graph, "email_graph_betweenness.gexf", version="1.2draft")
#Eigenvector centrality for email graph
eigenvector_cent_email = nx.eigenvector_centrality(email_graph)
key_max=0;
key_min=0;
maximum=0;
```

```
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
for key,value in eigenvector_cent_email.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value > 0.01 and value < 0.03 :
        email_graph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':30}
        second_group.append(value)
    if value > 0.03 and value < 0.05 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':40}
        third_group.append(value)
    if value > 0.05 and value < 0.07 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':50}
        fourth_group.append(value)
    if value > 0.07 and value < 0.1 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':60}
        fifth_group.append(value)
    if value > 0.1 and value < 0.2 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':70}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
```

```
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(sixth_group))
places_eigenvector={}
place_eigenvector=1
for i in sorted(eigenvector_cent_email.items(), key=lambda pair: pair[1]):
    places_eigenvector[i[0]]=place_eigenvector
    place_eigenvector+=1
print(eigenvector_cent_email.items())
nx.write_gexf(email_graph, "email_graph_eigenvector.gexf", version="1.2draft")
#PageRank centrality for email graph
pagerank_cent_email = nx.pagerank(email_graph)
key_max=0;
key_min=0;
maximum=0;
minimum=1;
key_max=0;
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
for key,value in pagerank_cent_email.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value > 0.001 and value < 0.002:
```

```
        email_graph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':30}
        second_group.append(value)
    if value > 0.002 and value < 0.003 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':40}
        third_group.append(value)
    if value > 0.003 and value < 0.005 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':50}
        fourth_group.append(value)
    if value > 0.005 and value < 0.007 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':60}
        fifth_group.append(value)
    if value > 0.007 and value < 0.01 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':70}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
print(key_min)
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(sixth_group))
places_pagerank={}
place_pagerank=1
for i in sorted(pagerank_cent_email.items(), key=lambda pair: pair[1]):
    places_pagerank[i[0]]=place_pagerank
    place_pagerank+=1
print(pagerank_cent_email.items())
nx.write_gexf(email_graph, "email_graph_pagerank.gexf", version="1.2draft")
#Load centrality for email graph
load_cent_email = nx.load_centrality(email_graph)
maximum=0;
minimum=1;
key_max=0;
```

```
key_min=0;
first_group=[];
second_group=[];
third_group=[];
fourth_group=[];
fifth_group=[];
sixth_group=[];
for key,value in load_cent_email.items():
    if value>maximum:
        key_max=key
        maximum=value
    if value<minimum:
        key_min=key
        minimum=value
    if value < 0.001:
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 255, 'a': 0}, 'size':10}
        first_group.append(value)
    if value > 0.001 and value < 0.004 :
        email_graph.node[key]['viz'] = {'color': {'r': 128, 'g': 128, 'b': 128, 'a': 0}, 'size':20}
        second_group.append(value)
    if value > 0.004 and value < 0.007 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 0, 'b': 255, 'a': 0}, 'size':30}
        third_group.append(value)
    if value > 0.007 and value < 0.03 :
        email_graph.node[key]['viz'] = {'color': {'r': 0, 'g': 100, 'b': 0, 'a': 0}, 'size':40}
        fourth_group.append(value)
    if value > 0.03 and value < 0.05 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 255, 'b': 0, 'a': 0}, 'size':50}
        fifth_group.append(value)
    if value > 0.05 and value < 0.071 :
        email_graph.node[key]['viz'] = {'color': {'r': 255, 'g': 0, 'b': 0, 'a': 0}, 'size':60}
        sixth_group.append(value)
print(maximum)
print(minimum)
print(key_max)
```

```
print(key_min)
print(len(first_group))
print(len(second_group))
print(len(third_group))
print(len(fourth_group))
print(len(fifth_group))
print(len(sixth_group))
places_load={}
place_load=1
for i in sorted(load_cent_email.items(), key=lambda pair: pair[1]):
    places_load[i[0]]=place_load
    place_load+=1
print(places_load)
nx.write_gexf(email_graph, "email_graph_loadcentrality.gexf", version="1.2draft")
#Summary measure
sum_places={}
degree_koeff=0.05
closeness_koeff=0.15
betweenness_koeff=0.1
eigenvector_koeff=0.2
pagerank_koeff=0.3
load_koeff=0.2
for key,value in load_cent_email.items():
    sum_places[key]=(degree_koeff * places_degree[key]) + (closeness_koeff * plac-
es_closeness[key]) + (betweenness_koeff * places_betweenness[key]) + (eigenvector_koeff * plac-
es_eigenvector[key]) + (pagerank_koeff * places_pagerank[key]) + (load_koeff * places_load[key])
for i in sorted(sum_places.items(), key=lambda pair: pair[1]):
    print(i)
```

Gnutella_graph.ipynb

```
import networkx as nx
gnutellaGraph=nx.read_edgelist('gnutella.txt', nodetype=int, create_using=nx.DiGraph())
#Degree centrality
degree_cent_gnutella= nx.degree_centrality(gnutellaGraph)
```

```
places_degree={}
place_degree=1
for i in sorted(degree_cent_gnutella.items(), key=lambda pair: pair[1]):
    places_degree[i[0]]=place_degree
    place_degree+=1
print(places_degree)
#Closeness centrality
closeness_cent_gnutella = nx.closeness_centrality(gnutellaGraph)
places_closeness={}
place_closeness=1
for i in sorted(closeness_cent_gnutella.items(), key=lambda pair: pair[1]):
    places_closeness[i[0]]=place_closeness
    place_closeness+=1
print(closeness_cent_gnutella.items())
#Betweenness centrality
betweenness_cent_gnutella= nx.betweenness_centrality(gnutellaGraph)
places_betweenness={}
place_betweenness=1
for i in sorted(betweenness_cent_gnutella.items(), key=lambda pair: pair[1]):
    places_betweenness[i[0]]=place_betweenness
    place_betweenness+=1
print(betweenness_cent_gnutella.items())
#Eigenvector centrality
eigenvector_cent_gnutella = nx.eigenvector_centrality(gnutellaGraph)
places_eigenvector={}
place_eigenvector=1
for i in sorted(eigenvector_cent_gnutella.items(), key=lambda pair: pair[1]):
    places_eigenvector[i[0]]=place_eigenvector
    place_eigenvector+=1
print(eigenvector_cent_gnutella.items())
#Pagerank centrality
pagerank_cent_gnutella = nx.pagerank(gnutellaGraph)
places_pagerank={}
place_pagerank=1
for i in sorted(pagerank_cent_gnutella.items(), key=lambda pair: pair[1]):
```

```
        places_pagerank[i[0]]=place_pagerank
        place_pagerank+=1
    print(pagerank_cent_gnutella.items())
    #Load centrality
    load_cent_gnutella = nx.load_centrality(gnutellaGraph)
    places_load={}
    place_load=1
    for i in sorted(load_cent_gnutella.items(), key=lambda pair: pair[1]):
        places_load[i[0]]=place_load
        place_load+=1
    print(places_load)
    #Summary measure
    sum_places={}
    degree_koeff=0.05
    closeness_koeff=0.15
    betweenness_koeff=0.1
    eigenvector_koeff=0.2
    pagerank_koeff=0.3
    load_koeff=0.2
    for key,value in degree_cent_gnutella.items():
        sum_places[key]=(degree_koeff * places_degree[key]) + (closeness_koeff * plac-
es_closeness[key]) + (betweenness_koeff * places_betweenness[key]) + (eigenvector_koeff * plac-
es_eigenvector[key]) + (pagerank_koeff * places_pagerank[key]) + (load_koeff * places_load[key])
    for i in sorted(sum_places.items(), key=lambda pair: pair[1]):
        print(i)
```

ВІДОМІСТЬ АТЕСТАЦІЙНОЇ РОБОТИ

Позначення	Найменування	Дод. відомості
	Текстові документи	
1	Пояснювальна записка	93 с.
2	Презентаційний матеріал	28 с.
	Інші документи	
3	Роздруківки програм	22 с.
4	Рецензія	2 с.
5	Відгук керівника	1 с.

					Методи аналізу та пошуку лідерів в соціальних мережах			
Змін	**Арк.**	**Номер докум.**	**Підп.**	**Дата**				
Розроб.		**Закутній С.В..**			**(Тема роботи) Відомість атестацій-ної роботи**		**Аркуш**	**Аркушів**
Перевір.		**Кіріченко Л.О.**						
Н. контр.		**Сидоров М.В.**				**ХНУРЕ кафедра ПМ**		
Затв.		**Тевяшев А.Д.**						