DSAA 5002 - Data Mining and Knowledge Discovery in Data Science

Final Exam Report – Q7 Social Media Network Analysis

50015940 Jiaxiang Gao

1 Data Load

1.1 Data Reading

The script reads edge data from a text file, 'socialmedia.graph.txt'. Each line is interpreted as an edge connecting two nodes in the network.

```
edges = []
# Read the data
with open('../Data_Q7/socialmedia.graph.txt', 'r') as file:
    for line in file.readlines()[2:]:
        node1, node2 = map(int, line.strip().split('\t'))
        edges.append((node1, node2))

```

1.2 Graph Construction

An undirected graph is constructed from the edge list.

```
# Create graph
g = Graph(edges=edges, directed=False)
g.delete_vertices(0)
```
在 2023.12.16 12:19:46 于 1ms内执行

2 Analyze the Network

2.1 Calculate and plot the clustering coefficient and degree distribution of the network

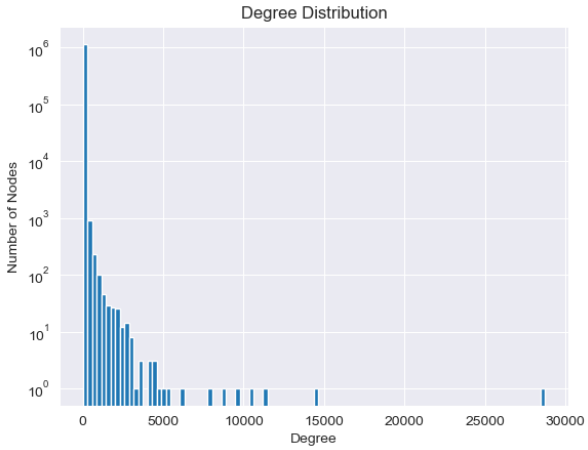
Clustering Coefficient Calculation: The average clustering coefficient computed is approximately 0.0062. This value indicates a low probability that two neighbors of a selected node are neighbors with each other, signifying a lack of tightly knit clusters within the network.

```
# Calculate the clustering coefficient
clustering_coefficient = g.transitivity_undirected()
print("Average Clustering Coefficient:", clustering_coefficient)
```
在 2023.12.16 12:19:49 于 1ms内执行

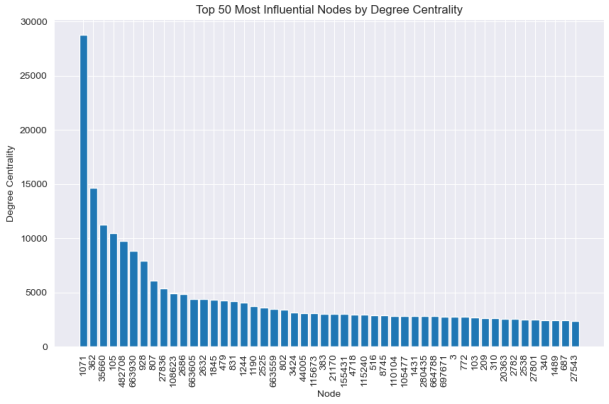
```
Average Clustering Coefficient: 0.006218559818028638
```

Degree Distribution:

The graph shows that the majority of nodes have a degree less than 1000, with the number of nodes decreasing rapidly as the degree increases. This kind of distribution is typical of social networks, where most people have a small number of connections while a few individuals (like celebrities or influencers) have a large number of connections. The long tail of the distribution suggests that there are a few nodes with a very high degree, although these are much less common.

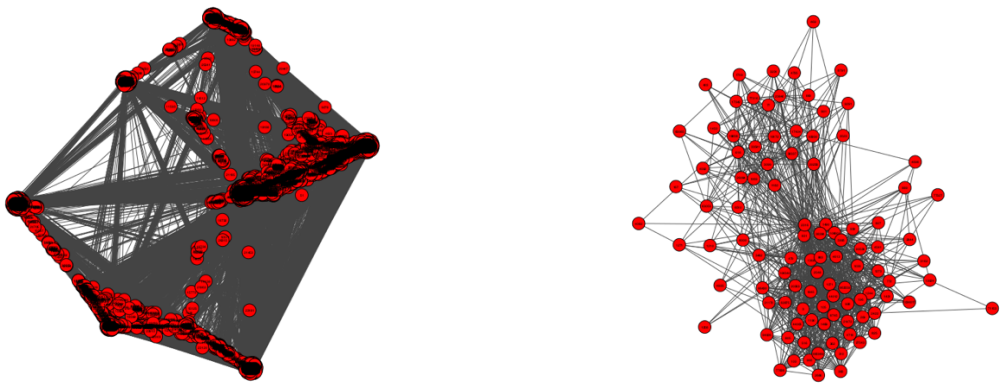


2.2 Identify the most influential nodes in a network and analyze them




```
In 9
# Most influential node
most_influential_node = top_nodes[0]

# Calculate the number of neighbors of the most influential node
num_neighbors = len(g.neighbors(most_influential_node))
print("Number of neighbors of the most influential node:", num_neighbors)

# Get the degrees of the neighbors of the most influential node
neighbor_degrees = g.degree(g.neighbors(most_influential_node))

# Calculate the average degree of the neighbors of the most influential node
average_neighbor_degree = sum(neighbor_degrees) / len(neighbor_degrees)
print("Average degree of the neighbors:", average_neighbor_degree)
在 2023.12.16 12:19:56 于 1ms内执行

Number of neighbors of the most influential node: 28754
Average degree of the neighbors: 40.92588857202476
```

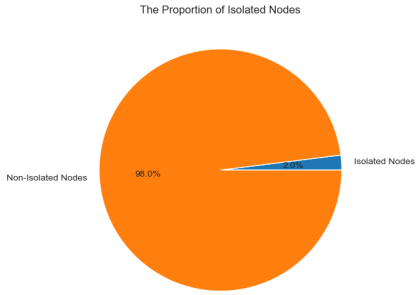
The node with the identifier 1071 is the most influential with 28,754 connections, significantly more than the others, indicating it is a major hub within the network. The degree centrality values drop from 28,754 for the most influential node to 2,380 for the 50th, showing a wide range in the influence levels even among the top nodes. The high degree of the top node suggests it plays a crucial role in the connectivity of the network, potentially acting as a bridge for many paths between other nodes.



The left figure shows the full subgraph of most influential node. The right figure shows the top 100 neighbors.

2.3 Identify Isolated Nodes in the Network

This network has a total of 22937 isolated nodes.




```python
# Identify the isolated nodes
isolated_nodes = [node for node, degree in enumerate(degrees) if degree == 0]
```
在 2023.12.16 12:21:19 于 13ms内执行

```python
# Print the number of isolated nodes and show 50 of them
num_isolated_nodes = len(isolated_nodes)
print("Number of Isolated Nodes:", num_isolated_nodes)
print("50 of them:", isolated_nodes[:50])

plt.figure(figsize=(6, 6))
plt.pie([num_isolated_nodes, len(degrees) - num_isolated_nodes], labels=['Isolated Nodes', 'Non-Isolated Nodes'], autopct='%1.1f%%')
plt.title('The Proportion of Isolated Nodes')
plt.show()
```
在 2023.12.16 12:21:22 于 44ms内执行

```
Number of Isolated Nodes: 22937
50 of them: [635473, 635474, 635475, 635476, 635477, 635478, 635479, 635480, 635481, 635482, 635483, 635484, 635485, 635488, 635489, 635490, 635491,
635493, 635494, 635495, 635496, 635498, 635499, 635501, 635502, 635503, 635505, 635506, 635512, 635513, 635514, 635515, 635517, 635518, 635519,
635520, 635521, 635522, 635523, 635524, 635525, 635527, 635528, 635529, 635531, 635532, 635533, 635534, 635535, 635536]
```

2.4 Recognize Connected Components in the Network

```python
# Get the all connected components
connected_components = g.connected_components()
```
在 2023.12.16 12:21:25 于 67ms内执行

```python
# Get the sizes of all connected components
num_components = len(connected_components)
print("The number of connected components:", num_components)

# Number of components with size > 1
num_nontrivial_components = sum([1 for component in connected_components if len(component) > 1])
print("The number of nontrivial connected components:", num_nontrivial_components)

# The size of the largest component
component_sizes = connected_components.sizes()
largest_component_size = max(component_sizes)
print("The size of the largest component:", largest_component_size)
```
在 2023.12.16 12:21:29 于 12ms内执行

```
The number of connected components: 22938
The number of nontrivial connected components: 1
The size of the largest component: 1134890
```

The network consists of 22,938 connected components. Out of these, only one connected component has more than one node in it, which is referred to as a non-trivial connected component. This implies that all other connected components consist of single, isolated nodes. The largest connected component consists of 1,134,890 nodes, which indicates that the vast majority of the network's nodes are part of this massive component.

2.5 Compute Average Shortest Path Length of the Network

Due to the large size of the original dataset, it takes a long time to calculate the average shortest path length and diameter. Therefore, I use a sampled (10%) dataset to calculate these metrics.

```python
# set the seed
random.seed(50015940)

# Set the sampling ratio
sampling_ratio = 0.1

# Choose a subset of edges from the original dataset randomly
sampled_edges = random.sample(edges, int(len(edges) * sampling_ratio))

# Create a sampled graph
sampled_graph = Graph(sampled_edges, directed=False)
```
在 2023.12.16 12:21:37 于 137ms内执行

The average shortest path length is a measure of the efficiency of information or transport flow in a network. It is calculated as the average number of steps along the shortest paths for all possible pairs of network nodes.

In the sampled graph, the average shortest path length is approximately 6.65. This value suggests that, on average, a node is about 6 to 7 steps away from another node, which is relatively low considering the size of the network and indicates a "small world" property commonly found in social networks.

```
# Calculate the average shortest path length of the sampled graph
sample_avg_shortest_path_length = largest_component_sampled.average_path_length()
print("The average shortest path length:", sample_avg_shortest_path_length)
```

在 2023.12.16 12:32:55 于 1ms内执行

```
The average shortest path length: 6.65350200406293
```

2.6 Calculate the Diameter of the Network

Same as Compute Average Shortest Path Length of the Network, here also use a sampled (10%) dataset to calculate these metrics.

The diameter of a network is the longest shortest path between any two nodes in the network. It is a measure of the linear size of a network and gives an idea of the maximum separation between nodes.

The calculated diameter is 28 for the sampled graph. This value means that the maximum number of steps required to travel between two nodes in the network is 28, which can be considered large.

```
# Calculate the diameter of the sampled graph
sample_diameter = largest_component_sampled.diameter()
print("The diameter:", sample_diameter)
```

在 2023.12.16 12:44:03 于 4ms内执行

```
The diameter: 28
```

2.7 Detect Community Structures in the Network

```
# community_multilevel
communities = g.community_multilevel()
```

在 2023.12.16 12:44:12 于 22ms内执行

```
# Number of communities
num_communities = len(communities)
print("Number of communities detected:", num_communities)

# Size of each community
community_sizes = communities.sizes()

# Average size of communities
avg_community_size = sum(community_sizes) / num_communities
print("Average size of communities:", avg_community_size)
```

在 2023.12.16 12:44:13 于 15ms内执行

```
Number of communities detected: 28950
Average size of communities: 39.994024179620034
```

The network has been found to have 28,950 communities.

The average size of these communities is approximately 40 nodes, suggesting that most communities in the network are relatively small.

Due to the large size of the original dataset, it takes a long time to visualize the communities. Therefore, we choose the communities with size between 20 and 100 to visualize.