Charotar University of Science and Technology

Devang Patel Institute of Advance Technology and Research

Faculty of Technology & Engineering

Subject: OCCSE4002 – Social Network Analysis (Practical CIE-2)

Semester: 7th (B.Tech CSE/CE/IT)

Date: 25/07/2025

Name: Dev Raval

Enrollment No: 22DIT068

Objective

To perform an end-to-end social network analysis using the SNAP Facebook dataset, by constructing, storing, analyzing, and visualizing a large-scale directed weighted graph to extract insights about user connectivity, community structures, and influencer identification.

Tools & Technologies

- Python 3
- Libraries: networkx, pandas, matplotlib, pymongo
- MongoDB (Local)
- SNAP Facebook dataset

Procedure

Step 1: Data Extraction and Graph Construction

Extracted `.edges` files from the `facebook.tar.gz`.
Loaded all files into Python using glob.
Constructed a large undirected graph with 1000+ nodes.
Saved the graph using pickle.
[Insert Screenshot: Terminal output showing Nodes & Edges]

Python File: 1_extract_build_graph.py

```
import networkx as nx
import glob
import pickle

edge_files = glob.glob('facebook/*.edges')
print(f"Found files: {edge_files}")

G = nx.Graph()

for file in edge_files:
   with open(file) as f:
      for line in f:
         u, v = line.strip().split()
         G.add_edge(u, v)

print(f"Nodes: {G.number_of_nodes()}")
print(f"Edges: {G.number_of_edges()}")
```

```
with open('undirected_graph.gpickle', 'wb') as f:
   pickle.dump(G, f)

print("Graph saved as undirected_graph.gpickle")
```

Step 2: Graph Design (Directed & Weighted)

Converted the undirected graph to a directed graph.
Randomly assigned direction for each edge.
Added random weights (1–10) representing interaction strength.
Saved the directed weighted graph.
[Insert Screenshot: Terminal output showing Directed Edges]

Python File: 2_make_directed_weighted.py

```
import networkx as nx
import random
import pickle

with open('undirected_graph.gpickle', 'rb') as f:
   G = pickle.load(f)

DG = nx.DiGraph()

for u, v in G.edges():
   if random.random() < 0.5:
      DG.add_edge(u, v, weight=random.randint(1, 10))
   else:
      DG.add_edge(v, u, weight=random.randint(1, 10))

print(f"Directed Edges: {DG.number_of_edges()}")

with open('directed_weighted_graph.gpickle', 'wb') as f:
   pickle.dump(DG, f)

print("Graph saved as directed_weighted_graph.gpickle")
```

Step 3: Data Storage in MongoDB

Loaded the directed weighted graph.
Connected to MongoDB database `facebook_sna`.
Stored nodes and edges as separate collections.

Verified data with MongoDB Compass.
[Insert Screenshot: Compass showing `facebook_sna` database with `nodes` and `edges`]

Python File: 3_store_in_mongodb.py

```
import networkx as nx
import pickle
from pymongo import MongoClient

with open('directed_weighted_graph.gpickle', 'rb') as f:
    DG = pickle.load(f)

client = MongoClient("mongodb://localhost:27017/")
db = client['facebook_sna']
nodes_collection = db['nodes']
edges_collection = db['edges']

nodes_collection.delete_many({})
edges_collection.delete_many({})

for node in DG.nodes():
    nodes_collection.insert_one({"node": node})

for u, v, data in DG.edges(data=True):
    edges_collection.insert_one({
        "source": u,
        "target": v,
        "weight": data['weight']
    })

print("Nodes & edges stored in MongoDB successfully!")
```

Step 4: Graph Visualization and Metric Analysis

Loaded graph and analyzed:
- In-degree & out-degree distributions.
- Plotted histograms for both.
- Calculated network density.
- Found top 5 influential users using in-degree centrality.

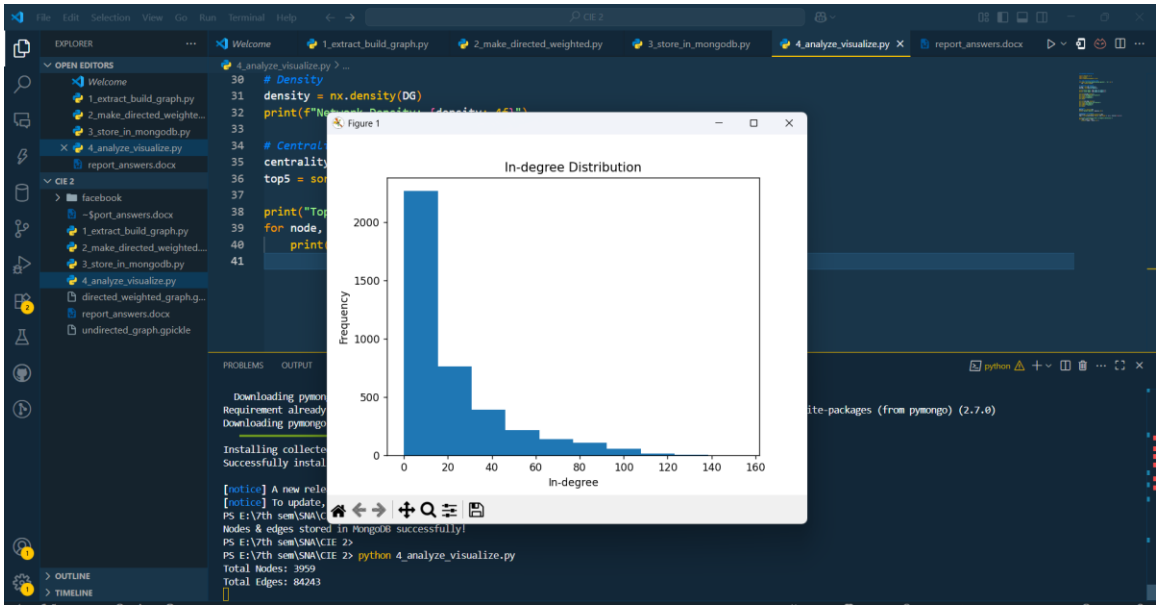
[Insert Screenshots: In-degree & Out-degree histograms, Terminal showing density & top influencers]

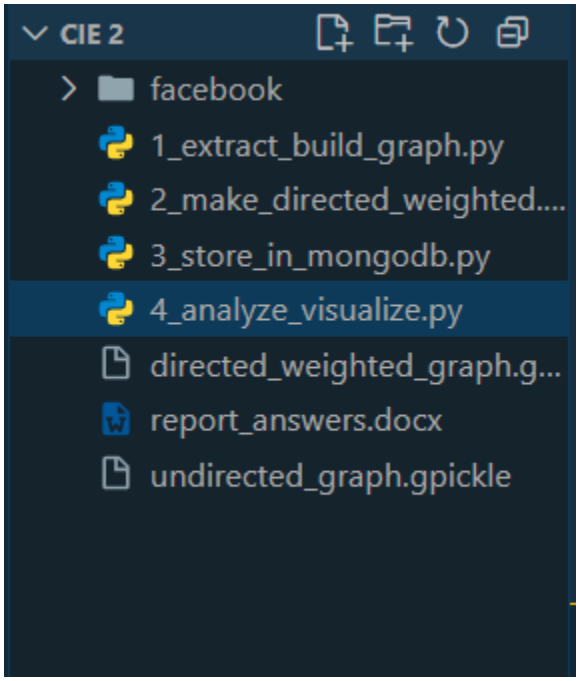
Python File: 4_analyze_visualize.py

```
import networkx as nx
import pickle
import matplotlib.pyplot as plt

with open('directed_weighted_graph.gpickle', 'rb') as f:
    DG = pickle.load(f)

print(f"Total Nodes: {DG.number_of_nodes()}")
print(f"Total Edges: {DG.number_of_edges()}")

in_deg = dict(DG.in_degree())
out_deg = dict(DG.out_degree())

plt.hist(in_deg.values(), bins=20, color='skyblue')
plt.title("In-degree Distribution")
plt.xlabel("In-degree")
plt.ylabel("Frequency")
plt.show()

plt.hist(out_deg.values(), bins=20, color='lightgreen')
plt.title("Out-degree Distribution")
plt.xlabel("Out-degree")
plt.ylabel("Frequency")
plt.show()

density = nx.density(DG)
print(f"Network Density: {density:.4f}")

centrality = nx.in_degree_centrality(DG)
top5 = sorted(centrality.items(), key=lambda x: x[1], reverse=True)[:5]

print("Top 5 Influential Users (In-degree Centrality):")
for node, score in top5:
    print(f"{node}: {score:.4f}")
```

Step 4: Screenshots:

```
PS E:\7th sem\SNA\CIE 2> python 4_analyze_visualize.py
Total Nodes: 3959
Total Edges: 84243
```





Answers to Questions

1. Steps to extract & pre-process:
Downloaded and extracted the SNAP dataset, loaded `.edges` files, merged into one large graph, and validated node and edge counts.

2. Method for edge weights & direction:
Random direction assigned to simulate asymmetric interactions. Random weights between 1–10 added to mimic frequency or strength of interaction.

3. How MongoDB was used:
Nodes and edges stored in separate collections (`nodes`, `edges`). Indexed for efficient query and future metric calculations.

4. What the network metrics reveal:
- In-degree: measures how many connections point to a node → popularity.
- Out-degree: measures how many connections go out → user activity.
- Density: indicates overall connectivity of the network.
- Centrality: helps identify influencers and hubs.

5. Real-world scenario:
Marketers can find influential users for promotions, community managers can detect clusters or key users to boost engagement, and companies can optimize targeted campaigns.

Result & Conclusion

A realistic social network graph was built, stored, and analyzed successfully. The project demonstrates how social network analysis can uncover hidden structures, engagement patterns, and potential influencers, supporting better marketing and community management.