Assignment 2 “Social Network Analysis”

Konstantina Marina Bletsa, AEM 243
Maria Karlaki, AEM 244

Part 1 - Agentic Social Network Analysis Chatbot

The project implements an agentic social network analysis system that allows interaction with a real social graph through natural language. The user can submit questions about the structure and properties of the network and receive answers based on classical metrics and graph analysis algorithms. The application was implemented in Python and utilizes agentic AI techniques, combining a Large Language Model.

The social network used in the work comes from the facebook_combined.txt file, which contains an undirected graph of Facebook social relationships, with each line of the file corresponding to an edge between two Facebook users. The nodes represent actual user IDs and not consecutive integers (i.e., it is not certain that every numerical ID exists in the graph, which is taken into account during queries and implementation).

The final architecture of the system is based on a single agent, which handles both communication with the user and the invocation of computational analysis tools. The agent uses an LLM to understand natural language and, depending on the query, selects and executes the appropriate tool.

The code is organized into 4 files. The graph_store.py file is responsible for loading the graph from the data file and keeping it in memory to avoid reloading it for each query. The tools_graph.py file contains all the social network analysis tools, implemented as Python functions. The agent.py file defines the agent, the language model used, and the agent's connection to the tools. Finally, the callback.py file records the conversations, storing the sessions in a json file for future reference.

Analysis Tools :

The graph_overview tool provides an overview of the social network structure. It calculates basic metrics such as the number of nodes and edges, the graph density, the number of connected components, and the size of the largest component. These metrics allow us to understand the size and connectivity of the network, as well as whether the network consists of a single dominant connected component or many isolated subnetworks. Indicative questions: "How many nodes and edges does the graph have?", "Give me a general overview of the graph", "Is the graph connected?", "What is the average degree and density of the network?"

The top_k_by_degree tool identifies the nodes with the highest degree, i.e., those with the most connections in the network. These nodes act as hubs and often play an important role in the dissemination of information and the cohesion of the network. Degree analysis helps identify particularly active or popular users within the social network. Suggested questions: "Which are the top 10 nodes with the highest degree?", "List the most connected nodes in the graph.", "Who are the main hubs of the network?"

The centralities_top_k tool calculates different measures of centrality, such as degree centrality, closeness centrality, betweenness centrality, and PageRank. Each measure captures a different notion of a node's "importance": for example, betweenness centrality shows nodes that act as bridges between different parts of the network, while PageRank highlights nodes that are connected to other important nodes. Through this analysis, we can understand which users are strategically important in the network. Indicative questions: "Which nodes are the most central according to degree centrality?", "Give the top nodes by betweenness centrality", "Which nodes are most important based on PageRank?", "Compare the most central nodes using different centrality measures".

The shortest_path tool calculates the shortest path between two nodes in the network, if it exists. The distance between two nodes expresses the minimum number of edges required to connect them. This metric is particularly useful for studying accessibility, social distance, and the speed at which information can spread between users. Sample questions: "Is there a path between node 4 and node 1000?", "Give the shortest path between node 4 and node 1000.", "What is the distance between node 10 and node 500?"

The ego_network tool analyzes the local subnetwork around a specific node, known as ego. It includes the node itself and its neighbors within a given radius. Through ego network analysis, we can study a user's immediate social neighborhood and the degree of cohesion of this microstructure. Example questions: "Analyze the ego network of node 4," "How many neighbors does node 1000 have in its ego network?", "Give statistics for the ego network of node 25 with radius 2."

The clustering_stats tool calculates statistics about the clustering coefficient of the network. This coefficient measures the tendency of nodes to form triangles, i.e., closed groups of friends. High clustering values indicate strong local cohesion, a characteristic of many social networks. Suggested questions: "What is the average clustering coefficient of the graph?", "Does the network show high local clustering?", "Give clustering statistics for the graph."

The k_core_summary tool analyzes the k-core structure of the network. A k-core is a subgraph in which each node has at least k connections within the subgraph. This analysis reveals the "hard core" of the network and helps identify particularly cohesive groups of users. Sample questions: "What is the size of the 10-core of the graph?", "What is the maximum core number in the network?", "Analyze the k-core structure of the graph."

The louvain_communities tool applies the Louvain algorithm to detect communities in the network. The algorithm maximizes modularity, i.e., the degree to which nodes are more densely connected within their communities than with the rest of the network. Community analysis allows us to understand the internal organization and social structure of the network. Sample questions: "Detect communities using the Louvain method," "How many communities does the graph have?", "What is the modularity of the network?", "Which are the largest communities in the graph?", "Describe the community structure of the network."

The degree_assortativity tool calculates the degree assortativity coefficient of the network. This metric shows whether nodes tend to connect to other nodes of similar degree. Positive assortativity indicates that high-degree nodes connect to other high-degree nodes, while negative assortativity indicates the opposite. Suggested questions: "Is the graph assortative or disassortative by degree?", "Compute the degree assortativity coefficient.", "Do high-degree nodes tend to connect to other high-degree nodes?"

The component_summary tool analyzes the connected components of the network and presents the sizes of the largest ones. This analysis helps to understand whether the network is fragmented or dominated by a large coherent structure. Suggested questions: "How many connected components does the graph have?", "What are the sizes of the largest connected components?", "Describe the component structure of the network."

The diameter_estimate tool calculates an approximation of the network diameter, i.e., the maximum distance between two nodes in the largest connected component. The diameter is an indication of the "size" of the network in terms of social distance and is related to phenomena such as the "small-world" effect. Suggested questions: "Estimate the diameter of the graph," "What is the approximate diameter of the largest connected component?", "How long is the longest shortest path in the network?"

The Neighbours Detection tool allows you to explore the immediate neighbors of a node in the network. First, it checks whether the given node ID is valid and whether the node exists in the graph. It then finds all nodes that are directly connected to that node (i.e., its immediate friends). If the number of neighbors is large, the code returns only a limited number of them so that the result remains manageable. It is also possible to request all neighbors without restriction. Along with the list of neighbors, information is returned about how many there are in total, how many appear in the result, and whether there are additional neighbors that do not appear. The user can use the tool with questions such as "Who are the neighbors of node 10?", "Show me the first 20 neighbors of node 5", "List all neighbors of node 42" or "How many neighbors does node 100 have?".

The Friends Recommendation tool implements a simple friend recommendation system based on the structure of the social network. For each node, it first identifies which other nodes are not already direct friends and considers them as candidates for recommendation. It then uses the Adamic Adar algorithm, which is based on mutual friends, to calculate how likely the user is to connect with each candidate node. Candidates are ranked based on this score and the best suggestions (top-k) are returned. For each suggested friend, additional information is provided, such as the number and a small sample of mutual friends, so that it is clear why that particular suggestion was made. The user can ask "Recommend friends for node 10" or "Suggest 6 new friends for node 0."

The Detection of Articulation Points and Bridges tool analyzes and identifies the most sensitive points of the graph, such as articulated nodes and bridge edges. Articulation nodes are those nodes which, if removed, cause the network to break into more pieces, while bridge edges are the connections that play the same role at the edge level. These elements are calculated only once and stored temporarily. The tool allows us to see either a specific number of these nodes or edges (top-k) or a summary of how many there are in total. The user can ask what the articulation points of the network are ("What are the articulation points of the graph?" or "List the top 5 articulation points"), what the bridge edges are ("What are the bridges of the graph?" or "List the top 10 bridges"), or request a summary of the situation ("Give me a summary of bridges and articulation points").

Run

The sna_graph_chatbot agent is executed using the adk web command in the terminal, which launches the ADK web environment, in the selection “select an agent” we choose sna_graph_chatbot provided that we are connected to the university's VPN. The user interacts with the system via a browser, submitting questions (such as those we suggested in the tools) in English and receiving answers based on the analysis tools. The lab's LLM is used via Ollama.

Part 2 - Simulation

The simulation implements the dissemination of information in a social network using an agent-based approach. Each node in the network corresponds to an agent, i.e., an individual, while the edges represent social relationships between individuals. At the beginning of the simulation, 60 agents are created and randomly assigned basic characteristics. Specifically, each agent has a binary variable "opinion," which represents their attitude or opinion, as well as a variable "informed," which indicates whether or not the agent is aware of the information being disseminated on the network. Initially, all agents are considered uninformed.

The social network structure is created using the Watts–Strogatz small-world model, which combines high local clustering with a short average path length between nodes. This model was chosen because it realistically approximates many real social structures. The user is asked to define two basic parameters of the model. The first is the parameter k, which determines the number of neighboring connections of each agent and therefore the density of the network. Higher values of k correspond to denser social networks, where agents have more social contacts. The second parameter is the reconnection probability p, which determines the degree of randomness in the network. As p increases, more local connections are replaced by random ones, creating distant connections that reduce the distances between agents and enhance the small-world effect.

When the network has been created, an agent is randomly selected as the initial carrier of the information, known as the seed agent. This agent is designated as informed and constitutes the starting point for dissemination. The simulation progresses in discrete time steps, the number of which is also determined by the user. At each time step, all uninformed agents examine their local social environment, i.e., their immediate neighbors in the network. If one of the neighbors is informed, the agent evaluates the probability of adopting the information.

The adoption decision is not random, but is based on a simple rule of behavior that incorporates the concept of homophily. Specifically, the probability of an agent becoming informed increases as the percentage of informed neighbors who share the same opinion increases. In this way, the simulation captures the phenomenon whereby individuals tend to be more influenced by social circles with similar attitudes or beliefs.

During the simulation, key metrics are recorded to evaluate the spread. One of the most important is diffusion reach, which is the percentage of agents who have been informed at each time step. This parameter allows the spread of information over time to be monitored and phenomena such as slow initial adoption, rapid increase (cascade), and final saturation to be identified. In addition, the network's homophily index is calculated, which expresses the percentage of edges connecting agents with the same opinion and is used for the qualitative interpretation of the results.

The simulation results are visualized in two ways. First, a graph of the spread of information over time is presented, showing the diffusion reach at each step of the simulation. Next, the final graph of the social network is displayed, in which the nodes represent the agents and the edges represent the social relationships, while the color of the nodes indicates whether the agents are informed or not at the end of the process. These representations make it possible to understand both the dynamics of diffusion and its relationship to the structure of the network.

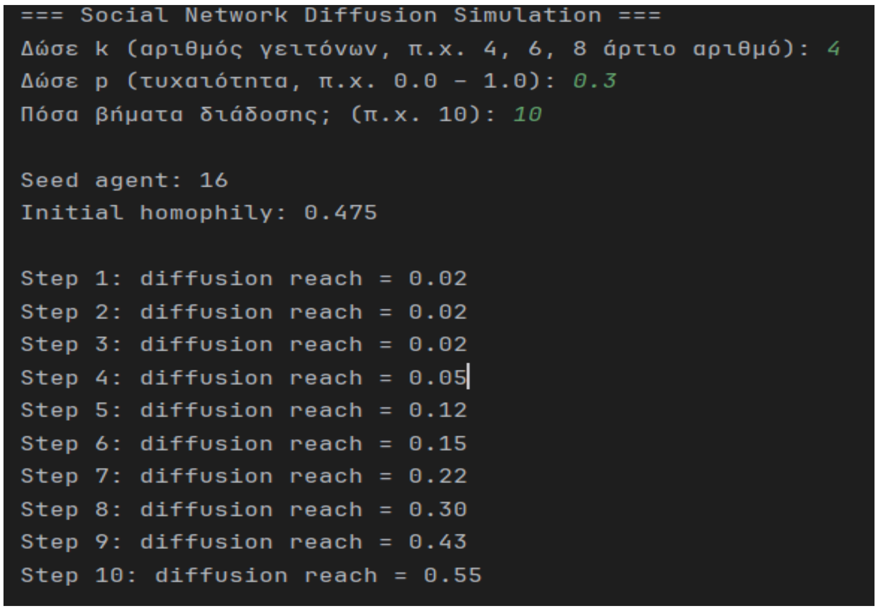
Running the Simulation

To run the simulation, we press run in the main file and within the environment in which we run our code in the terminal, the message "Give me k, the number of neighbors must be even" appears, and we give whatever number we want, then it asks "Enter the value p" which corresponds to the degree of randomness of the network and finally "Enter the number of simulation iterations". When we fill them in, the results will appear in the run environment and two windows with diagrams.

Simulation Results and Observations

In this particular simulation example, the parameters k = 4 and p = 0.3 were used in a social network of 60 agents. The value k = 4 means that each agent has a relatively limited number of contacts-neighbors, while the value p = 0.3 introduces a moderate degree of randomness into the network structure. This combination corresponds to a typical small-world network with relatively low density. The initial information carrier (seed agent) was agent 16, which was selected at random. The initial value of the network's homophily is 0.475, which indicates that the network does not exhibit strong homophily; social relationships do not connect agents with the same opinion in the majority of cases (low homophily does not help the spread of information).

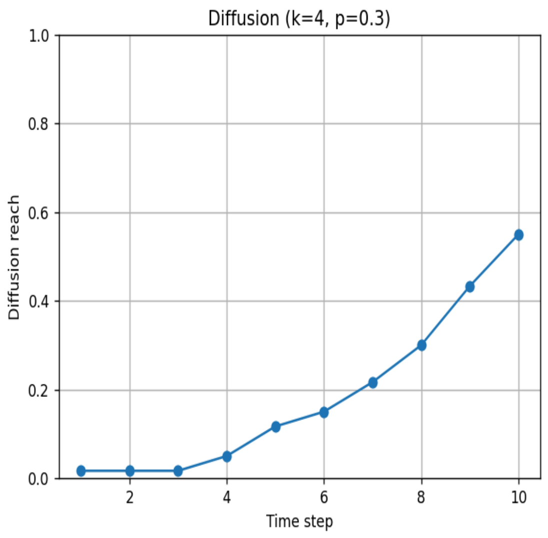
In the early stages (steps 1 to 3), diffusion reach remains almost zero (approximately 2%), indicating that information remains confined to the very local environment of the seed agent. At this stage, there are not enough social "bridges" to allow for wider dissemination, while low homophily limits the likelihood of adoption by neighbors. From step 4 onwards, diffusion increases, with diffusion reach rising to 5% and then to 12% and 15%, indicating that the information is beginning to spread beyond the initial core and into neighbouring areas of the network. This phenomenon is directly related to the existence of random connections due to the value p = 0.3. In the next steps of the simulation (steps 7 to 10), the diffusion accelerates significantly. The diffusion reach increases from 22% to 30%, then to 43%, and finally reaches 55%. This behavior is a typical example of a cascade phenomenon, where once a critical threshold of informed agents is exceeded, information begins to spread more rapidly across the network. However, the spread is not universal, as approximately 45% of agents remain uninformed at the end of the simulation. The graph shows that the red nodes (informed agents) form several clusters, but do not cover the entire graph. The gray nodes mainly correspond to agents located on the periphery of the network or in areas with fewer connections and low homophily with their informed neighbors. This shows that the information did not manage to fully penetrate all the social infrastructures of the network.

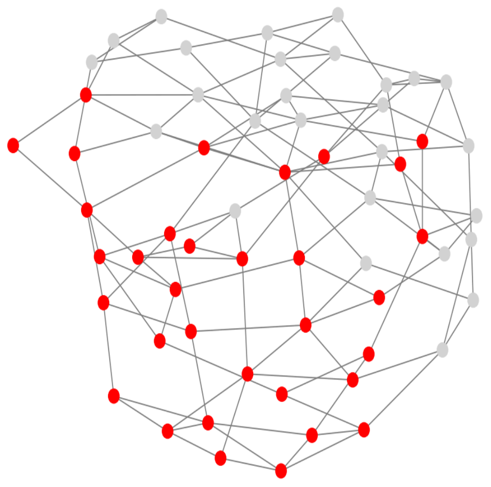


```
=== Social Network Diffusion Simulation ===
Δώσε k (αριθμός γειτόνων, π.χ. 4, 6, 8 άρτιο αριθμό): 4
Δώσε p (τυχαιότητα, π.χ. 0.0 - 1.0): 0.3
Πόσα βήματα διάδοσης; (π.χ. 10): 10

Seed agent: 16
Initial homophily: 0.475

Step 1: diffusion reach = 0.02
Step 2: diffusion reach = 0.02
Step 3: diffusion reach = 0.02
Step 4: diffusion reach = 0.05
Step 5: diffusion reach = 0.12
Step 6: diffusion reach = 0.15
Step 7: diffusion reach = 0.22
Step 8: diffusion reach = 0.30
Step 9: diffusion reach = 0.43
Step 10: diffusion reach = 0.55
```





The overall simulation study shows that the structure of the social network plays a decisive role in the dynamics of information dissemination. The density of the network, as expressed by the parameter k, directly affects the speed and extent of dissemination. Networks with a larger number of social connections per agent facilitate faster information dissemination and lead to higher levels of final coverage.

At the same time, the degree of randomness of the network, which is controlled by the parameter p, proves to be critical for the occurrence of cascade phenomena. Even a limited number of random connections is sufficient to create bridges between different local communities, reducing distances in the network and allowing information to penetrate areas that would otherwise remain isolated.

Homophily emerges as an important mechanism for filtering information. When social relationships mainly connect agents with similar views, information spreads more easily within homogeneous groups, but finds it difficult to pass on to different communities. In contrast, in networks with low homophily, dissemination is slower and often remains partial, even if there are random connections.

The code for this project can be found in the Github repository at the following link:

https://github.com/kbletsa/Social-Network-Analysis-Project/tree/main