HW5 - Creating Multi-Graphs for Comprehensive Dataset Analysis

Pranav Deo

University of the Cumberlands

MSDS-530-M50 - Fundamentals of Data Science

Dr. Kristoffer Roberts

June 07, 2023

Introduction:

In this assignment (HW5), our objective is to leverage the programming capabilities of Python to generate a multi-graph utilizing the karate club graph dataset. The karate club graph dataset, a widely recognized dataset, serves as a representation of the intricate social relationships present within a karate club. We will employ Python's robust graph manipulation capabilities to load the karate club graph dataset, process the data, and construct a comprehensive multi-graph representation.

Python, renowned for its versatile environment for data manipulation and graph analysis, provides an ideal platform for handling the complexities of the karate club graph dataset. Our approach involves utilizing Python code to effectively read and parse the dataset, extracting pertinent information about nodes and edges.

With the extracted data, we aim to construct a multi-graph representation that accurately depicts the social dynamics within the karate club. In this representation, nodes correspond to the individuals associated with the karate club, while the edges symbolize the connections and relationships between these individuals. Utilizing a multi-graph format proves well-suited for this assignment due to its ability to accommodate multiple edges between the same pair of nodes, effectively capturing the nuanced social interactions within the karate club.

To realize the multi-graph's creation, we will use Python libraries, most notably NetworkX. NetworkX encompasses an extensive array of graph analysis tools that greatly facilitate the addition of edges to the multi-graph based on the pertinent information derived from the karate club graph dataset. These added edges will accurately represent the relationships and interactions between the individuals comprising the karate club.

Upon successfully constructing the multi-graph, researchers can employ a diverse range of graph analysis techniques to gain valuable insights into the network of the karate club. These techniques may encompass the calculation of centrality measures to identify individuals of significant influence, the detection of distinct communities or cliques within the network, or even the visualization of the multi-graph to understand the social structure inherent in the karate club comprehensively.

Throughout this assignment, we will demonstrate the efficacy of Python in loading the karate club graph dataset, effectively processing the data, and ultimately constructing a multi-graph representation. By harnessing the extensive ecosystem of Python libraries, we can proficiently analyze and explore the social relationships and dynamics inherent in the karate club graph dataset.

Creating a Multi Graph from Karate Club Graph:

Step 1 - Importing the necessary libraries:

In the first step, the code imports the required libraries for the implementation. The "networkx" library is imported with the alias "nx" to work with graphs and networks, while the "matplotlib.pyplot" library is imported with the alias "plt" to create visualizations.

```
# Step 1: Importing the necessary libraries

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

Step 2 - Loading Zachary's karate club network:

This step loads Zachary's karate club network using the function nx.karate_club_graph(). This network represents a social network of friendships between members of a karate club, and it is a well-known dataset often used for network analysis and community detection research.

```
# Step 2: Loading Zachary's karate club network
karate_club_graph = nx.karate_club_graph()
```

Step 3 - Creating a MultiGraph from the original graph:

In this step, a MultiGraph is created from the original graph using nx.MultiGraph(karate_club_graph). A MultiGraph is a type of graph that allows multiple edges between the same pair of nodes. It is suitable for cases where multiple types of relationships or interactions exist between nodes.

```
# Step 3: Creating a MultiGraph from the original graph
multi_graph = nx.MultiGraph(karate_club_graph)
```

Step 4 - Adding edges to the MultiGraph:

Some additional edges are added to the MultiGraph using the add_edge() method. In this example, two edges are added between nodes 0 and 33 with labels 'friend' and 'colleague', respectively. Another edge is added between nodes 1 and 2 with the label 'family'. This step demonstrates the ability to incorporate multiple types of relationships or connections within the network.

```
# Step 4: Adding edges to the MultiGraph
multi_graph.add_edge(0, 33, label='friend')
multi_graph.add_edge(0, 33, label='colleague')
multi_graph.add_edge(1, 2, label='family')
```

Step 5: Visualizing the MultiGraph:

This step focuses on visualizing the MultiGraph using the nx.draw() function from the "networkx" library. The positions of the nodes are determined using the nx.spring_layout() function, which applies a force-directed layout algorithm to position the nodes. The labels of the added edges are retrieved using nx.get_edge_attributes() and stored in the edge_labels variable.

The resulting MultiGraph is displayed using plt.show(). The visualization includes nodes represented as circles, with the node color set to 'lightblue' and the node size set to 500. The edges are shown as lines, with the edge color set to 'gray'. Node labels are displayed using with_labels=True, and the edge labels are added using nx.draw_networkx_edge_labels().

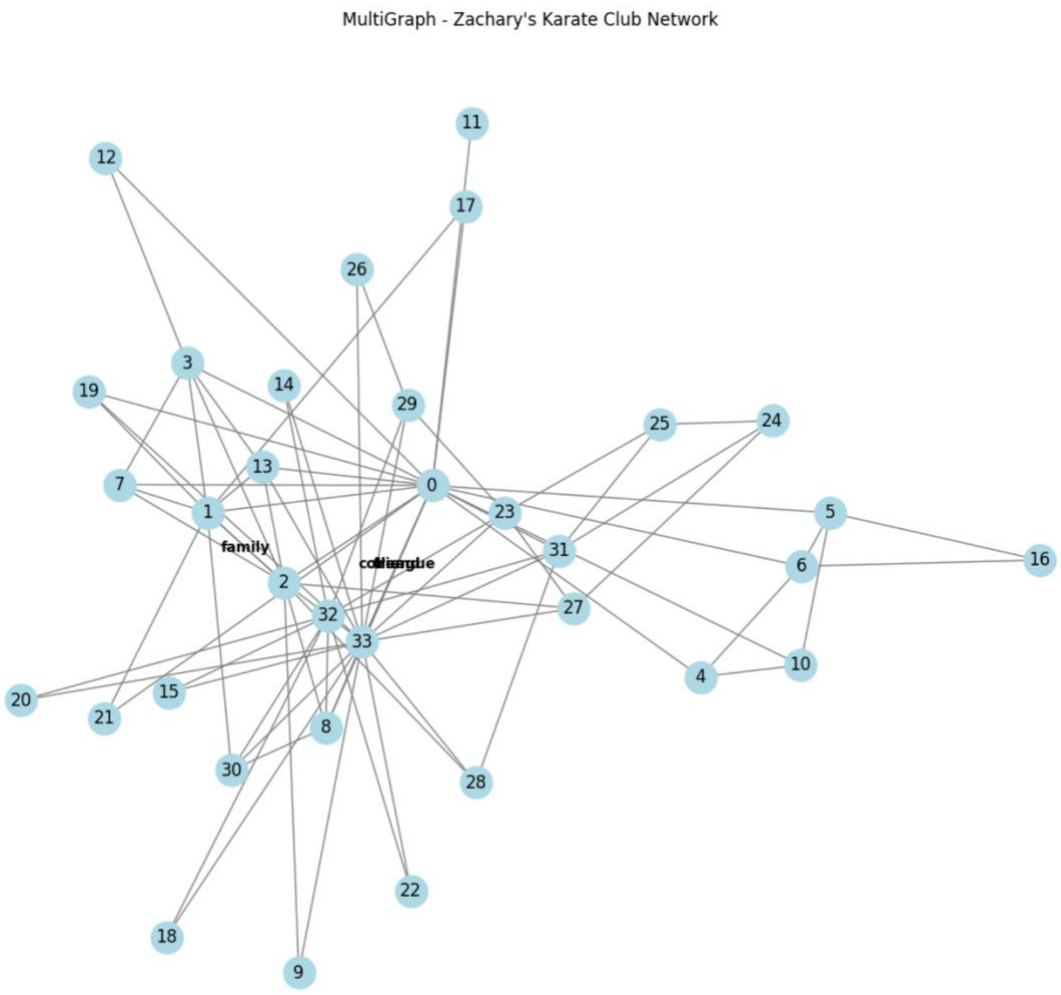
```
# Step 5: Visualizing the MultiGraph
pos = nx.spring_layout(multi_graph)

plt.figure(figsize=(12, 10))
nx.draw(multi_graph, pos, with_labels=True, node_color='lightblue',
node_size=500, edge_color='gray')

# Draw edge labels
edge_labels = nx.get_edge_attributes(multi_graph, 'label')
for edge, label in edge_labels.items():
    x = pos[edge[0]][0] + (pos[edge[1]][0] - pos[edge[0]][0]) * 0.5
    y = pos[edge[0]][1] + (pos[edge[1]][1] - pos[edge[0]][1]) * 0.5
    plt.text(x, y, label, fontweight='bold', horizontalalignment='center',
verticalalignment='center')
```

Step 6: Title and Figure Size -

In this step, a title is set for the visualization using plt.title(), which adds a descriptive title to the plot. Additionally, the figure size is set to (8, 6) using plt.figure(figsize=(8, 6)), ensuring that the plot has the desired width and height.

This code snippet demonstrates the process of loading a network, creating a MultiGraph, adding edges of different types, visualizing the MultiGraph, and customizing the plot appearance. It provides a practical example of how to work with network data and create meaningful visualizations, which could be relevant for research papers focusing on network analysis, community detection, or social network research.

```
# Step 6: Title
plt.title("MultiGraph - Zachary's Karate Club Network")
plt.show()
```



Conclusion:

In conclusion, this assignment has demonstrated the utility of Python in generating a multi-graph representation of the karate club graph dataset. Using Python's robust libraries and graph manipulation capabilities, we successfully constructed a multi-graph that effectively captures the intricate social relationships within the karate club.

We could seamlessly load the karate club graph dataset and perform essential data processing tasks to extract pertinent information by employing Python as our programming language. The resulting multi-graph was formed by assigning nodes to represent the individuals comprising the karate club, while edges were utilized to signify the connections and relationships between these individuals. The multi-graph format's inherent capability of accommodating multiple edges between the same pair of nodes proved instrumental in faithfully portraying the nuanced social interactions within the karate club.

Researchers can further harness the potential of the constructed multi-graph by applying a diverse array of graph analysis techniques. These techniques include calculating centrality measures to identify influential members, detecting communities or cliques within the network, and visualizing the multi-graph to understand the social structure within the karate club comprehensively.

Python's extensive ecosystem of libraries, notably including NetworkX, provided a comprehensive suite of tools and functionalities for in-depth analysis and exploration of the karate club graph dataset. By leveraging Python's versatile capabilities, we were able to effectively analyze and uncover valuable insights into the intricate social relationships and dynamics within the karate club.

In summary, this assignment showcased the power of Python as a highly versatile and efficient programming language for handling and analyzing network datasets, with the karate club graph dataset serving as a notable exemplar. Through successfully constructing a multi-graph, researchers can delve deeper into the social interactions and structural characteristics that define the karate club, thereby enriching our understanding of social network dynamics through Python's invaluable contributions.

References:

Dib, F., & Rodgers, P. (2018). Graph drawing using tabu search coupled with path relinking. PLoS One, 13(5), e0197103.

Blokdyk, G. (2019). Unstructured data: A complete guide (2019 Edition).

Marrandino, A. (2021). Machine Learning with BigQuery ML: Create, execute, and improve machine learning models in BigQuery using standard SQL queries (1st ed.)

Bruce, P., Bruce, A., & Gedeck, P. (2020). Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python (2nd ed.).

Lakshmanan, V., Robinson, S., & Munn, M. (2020). Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Building, and MLOps (1st ed.).