**News network dataset**

1. Loading and understanding the dataset (attached notebook)
2. Visualization tasks
3. Calculate and Compare Different Centrality Measures:

* Task: Compute various centrality measures for all nodes in the network and compare their distributions and rankings.
* Steps:
  + Use the networkx library to calculate measures like:
    - Degree Centrality: Measures the number of direct connections a node has.
    - Betweenness Centrality: Measures how often a node lies on the shortest path between other nodes.
    - Closeness Centrality: Measures how quickly a node can spread information through the network.
    - Eigenvector Centrality: Measures the influence of a node based on the centrality of its neighbors.
    - Also calculate other centrality meaures like PageRank and Katz.
  + Store these centrality values in a pandas DataFrame or dictionary, associated with the sentence IDs.
  + Visualize the distribution of each centrality measure (e.g., using histograms).
  + Identify the top-k most central nodes based on each measure and compare these lists. Discuss why different measures might identify different nodes as most central.

1. Community detection:

* Task: Apply various community detection algorithms to the network and compare the resulting partitions.
  + Identifying Thematic Clusters within a Single Article: Apply community detection algorithms to the sentence-level graph of a single news article. This can reveal sub-topics or different angles discussed within the same article. Sentences within the same community likely cover a similar specific aspect of the overall news story.
  + Grouping Similar Articles Across the Dataset: Build a graph where each node represents a news article and edges represent the similarity between articles (calculated using TF-IDF or other methods on the article text). Apply community detection to this article-level graph. The identified communities will consist of groups of news articles that are similar in content, allowing you to discover clusters of articles covering the same or related news events or topics.
* Steps:
  + Use libraries like networkx such as:
    - Louvain method: An iterative algorithm that optimizes modularity.
    - Leiden method
  + Run several of these algorithms on your network.
  + Obtain the community assignments for each node.
* Task: Analyze the Properties of Identified Communities: such as their size, density, and internal structure.
* Steps:
  + Calculate the number of nodes and edges in each community.
  + Compute the density of each community (ratio of existing edges to possible edges).
  + Visualize individual communities or the network with nodes colored by their community membership
  + Analyze the distribution of community sizes.

* Task: Use metrics to assess the quality of the community structures found by different algorithms.
* Steps:
  + Calculate modularity: A widely used metric that measures how well a network is partitioned into communities. Higher modularity generally indicates a better partition.
  + Calculate NMI
  + Calculate ARI

1. Link Prediction:

Identifying Potentially Similar Sentences within an Article:

* Task: In a sentence-level graph where an edge exists if the similarity is above a certain threshold, link prediction can identify pairs of sentences whose similarity is below the threshold but is predicted to be high based on the graph's structure (e.g., they share common neighbors, or paths exist between them).
* Steps:
  + Pre-process the dataset into train-test data.
  + Use link prediction methods based on network structure (e.g., common neighbors, Jaccard coefficient, Adamic-Adar index, preferential attachment).
  + Evaluate the performance of your prediction model using appropriate metrics (e.g., Area Under the ROC Curve (AUC), Average Precision (AP)).
* Task: Similarly predict similar articles across the dataset.

1. Graphlets:

* Task: Identify graphlets.
  + Focus on smaller graphlets initially, such as 2-node graphlets (an edge), 3-node graphlets (a path and a triangle), and potentially some 4-node graphlets (using ORCA tool).
* Task: Compute Graphlet Frequency and Distribution. Count the occurrences of different graphlets in your network to understand its local structure.
  + Count the number of times each defined graphlet appears in your network.
  + Calculate the frequency of each graphlet type.
  + Visualize the distribution of graphlet counts.
* Task: Use Graphlet Features for Downstream Tasks:
  + Once you have the graphlet orbit count vectors for each node, use these vectors as features in a machine learning model.
  + For example, train a classifier to predict protein properties (as in node classification) using the graphlet features instead of or in addition to node features (after you have studied GNNs).
  + Apply clustering algorithms to the nodes using their graphlet feature vectors to see if similar graphlet patterns correspond to functional or structural groupings.

1. GNNs: Given that you've represented your news articles as graphs (at the article level across the dataset) and have associated features (like TF-IDF vectors) with the nodes, you perform:

* Steps:
  + Define a target variable: The labels for your node classification task are the news categories ('politics', 'crime', 'sports', 'movies'). Convert these categorical labels into numerical format (e.g., 0 for 'politics', 1 for 'crime', etc.) using techniques like Label Encoding or one-hot encoding. Store these numerical labels in a NumPy array aligned with your article nodes.
  + Split the nodes into training, validation, and test sets.
  + Build a simple GNN model (e.g., a Graph Convolutional Network - GCN) that takes node features and the graph structure as input.
  + Train the GNN model to predict the target variable for each node.
  + Evaluate the model's performance using metrics appropriate for classification.

1. Perform the above classification task using:
   * Graph Attention Networks
   * Graph Transformers