I Test Assignment: Political Blog Classification using Graph Convolutional Networks

1 Task Overview

The project focuses on classifying political blogs as either "liberal" or "conservative" by leveraging Graph Convolutional Networks (GCNs). GCNs are particularly well-suited for this task as they can capture and learn from the interconnected nature of data, such as the relationships between blogs based on hyperlinks or shared topics.

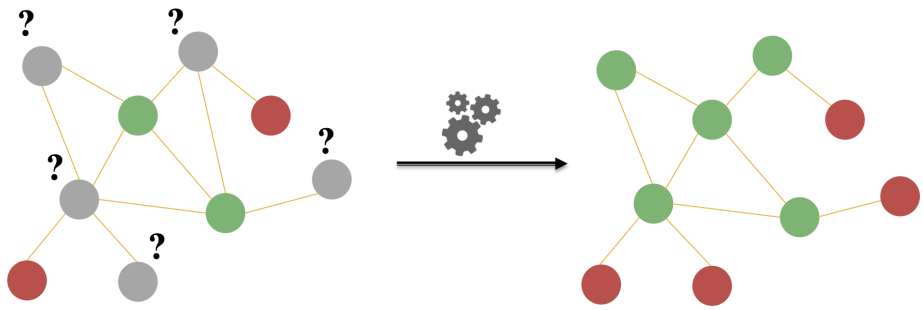

Figure 1: Example of the node classification task

Node classification is a machine learning task where you aim to predict labels (or properties) for individual nodes within a graph. Imagine a social network where nodes are users and edges represent friendships. Node classification could involve predicting a user's profession (lawyer, doctor, etc.) based on their connections and potentially some profile information.
Graph Convolutional Networks (GCNs) are a powerful tool for tackling node classification tasks. Unlike traditional CNNs that operate on grids (like images), GCNs are designed to work on graphs, leveraging the relationships between nodes.
Moreover, GCN models allow practitioners to consider feature representation for each node and edge in the graph. These features could be things like the content of a post on a social network (e.g., in the form of text embeddings) or the number of publications written by an author in a citation network.
Note: The solutions that make use of text embeddings of the content extracted by blogs, will be evaluated more. Pay attention: the dataset does not provide such content, and the students, if they want to exploit this information, have to extract it by themselves.

2 Objective

The main objective is to develop a robust GCN-based model that can effectively classify political blogs using the Polblogs dataset. This dataset comprises political blogs and their corresponding links, providing a network structure that can be exploited by GCNs.

Sub-Objectives

- Load the **Polblogs** dataset and understand its structure.

- Construct a graph representation of the data, where nodes represent blogs, and edges represent hyperlinks between them.
- Create an adjacency matrix to represent the graph structure.
- **Optional** Extract textual content from each blog (optional).
- **Optional** Generate text embeddings using transformer-based techniques for the extracted text.
- Implement and train a Graph Neural Network model or other adequate model for node classification.
- Evaluate the model's performance using appropriate metrics.
- Optimize the model for improved accuracy and robustness.

3 Dataset

For this Project assignment, the dataset will be a directed network of hyperlinks between weblogs on US politics, recorded in 2005 by Adamic and Glance: `https://websites.umich.edu/~mejn/netdata/polblogs.zip`
L. A. Adamic and N. Glance, "The political blogosphere and the 2004 US Election", in Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem (2005).
The polblogs dataset is a network representing the political blogosphere leading up to the 2004 US election. It's designed for studying how blogs connect and form communities.
Here's a breakdown of its structure:

- **Nodes**: Each node represents a single blog. There are around 1,490 blogs in the dataset.
- **Edges**: Directed edges connect nodes, indicating a hyperlink from one blog to another. The dataset contains roughly 19,025 edges.
- **Node Attributes**: Each blog (node) has a label indicating its political leaning: liberal or conservative.
- **Structure type**: The polblogs dataset represents a directed network.
- **Data format**: The specific format might vary depending on where you download it from, but common options include GML, DyNetML, and possibly plain text.

4 Model Architecture

The core of this project is the implementation of a GCN model. You can consider a multi-layer GCN architecture. Each layer in the GCN performs a message-passing operation, where nodes aggregate information from their neighbours based on the adjacency matrix and feature vectors. The output of the final layer is fed into a classification layer, which uses a softmax function to output probabilities for "liberal" or "conservative".
The students can choose the overall architecture, including edge embeddings based on text embeddings, but the overall complexity must not be lower than a GCN. More complex architectures are welcomed, if they properly fit the task.

5 Training and Evaluation

Train the model on the training set using suitable loss functions and optimization techniques. Monitor the training process by evaluating the model's performance on the validation set. Use appropriate evaluation metrics. Fine-tune the model hyperparameters to improve its performance on the validation set. Finally, evaluate the trained model on the test set and report its performance metrics.

6 Model Analysis

Perform an analysis of the model's performance. Identify any challenges or limitations faced by the model. Provide recommendations for further improvements or modifications to enhance the model's accuracy and efficiency.

7 Deliverables

- A well-commented Python code implementation of the models.
- A report documenting the detailed approach taken to solve the problem, including data preprocessing, model architecture, training, and evaluation results.
- Analysis and discussion of the model's performance, challenges faced, and recommendations for improvement.
- A presentation that will be discussed at the time of the Oral Exam.

Notes

You are encouraged to utilize existing deep learning libraries such as TensorFlow or PyTorch for implementing the model. Make sure to properly document your code and provide clear explanations in the report to demonstrate your understanding of the concepts and techniques used.