Citation Network

Project Scope and Objectives

Problem Statement:

NLP methods that exist traditionally classify papers solely on content ignoring rich interconnections that exist between them. Our project explores how graph based learnings can often outperform these and how to set up a fully functioning pipeline to integrate, deploy and reproduce the outcome.

Project Objectives and expected impact:

- To build a reproducible ML pipeline that can classify nodes in a citation network
- Use graph based models like GAT, GCN for node classification
- To integrate open source tools like Pytorch Geometric and MLflow into the workflow
- To track experiments and version control collaboratively using git, DVC etc.

Success Metrics:

The success metrics used in this project will be divided into two parts. The first part is the success metrics for the model training. Here the selected success metrics are negative log likelihood loss and accuracy. The negative log likelihood is good for classification problem such as this project. The second part is the CI/CD pipeline. Here we will be looking for reproducibility which means the entire pipeline will be reproducible and reliability which means the system will be reliable and have fault tolerance.

Description:

Citation networks have been found to prove meaningful relations between documents in academic research papers. These relations often indicate similarities in topics, which are often ignored by traditional text classification models. Our project aims to exploit this relation structure using Graph Neural Networks specifically GCN and GAT.

We have chosen the Cora dataset, which consists of 2708 papers, each described by a sparse bag-of-words vector and labeled under one of seven categories. Edges represent the citation relation between papers. This graph structure is well suited for GNN’s which update each node’s representation by aggregation of features from its neighbours. We will work on both GCN and GAT.

Our core framework would be Pytorch Geometric, an open source library that simplifies GNN implementations on Pytorch/Keras. It also provides built-in support for full batch training on citation networks, model wrappers for GCN and GAT and easily integrates with existing ML workflows.

To track model configurations and results we will use mlflow. Our project repository will follow the MLOps cookie cutter template for organizing code, data, configs and logs so that they can be reproduced easily. Environment dependencies will be specified in a requirements.txt file and training will be done locally or on Colab.

By the end of Phase 1 we aim to produce a working GCN based classification model, and a reproducible pipeline ready for scaling in later phases.

Selection of Data:

Dataset chosen and Justification:

We have chosen the Cora dataset as it is a standard benchmark for graph based learning models. It contains 2708 ml research papers which are classified into 7 topic categories with over 5429 citation edges. Each paper has a 1433 dimensional binary feature vector that indicates the presence or absence of common words from bag-of-words. The dataset is also publicly available.

Dataset source and access method:

The Cora dataset is part of the pytorch geometric library and can be easily downloaded and loaded using the same library. Currently the access method for the data is local, with plans to move it to some cloud space by next phase.

Preprocessing steps:

The Cora dataset is a completely cleaned and well-organized dataset. The only preprocessing being applied is normalizing data, which normalizes all the features.

Model Considerations:

Model architectures considered:

We have considered two architectures: Graph Convolutional Networks and Graph Attention Networks.

Rationale for model choice:

Currently we have already trained Graph Convolutional Networks which is giving up to 79% accuracy on the test subset of the Cora data. But the attention mechanism of Graph Attention Networks are shown to learn significantly more important features and correlation using the attention mechanism. Hence, by the next phase we might switch to Graph Attention Networks and compare the results.

Source/Citation for any pre built models:

We haven't used any pre-built models yet. We have built our own architecture for the Graph Convolutional Network.

Open-source Tools

Third-party packages:

Core ML:
PyTorch Geometric : Extension of PyTorch for handling graph-structured data. Used for implementing Graph Neural Networks (GNNs) for citation network analysis.

Data Processing & Analysis
NetworkX: Used for graph manipulation and analysis. Helps in processing citation networks, computing graph metrics, and visualizing network structures.
scikit-learn: Provides tools for data preprocessing, model evaluation, and traditional machine learning algorithms. Used for feature engineering and baseline models.

Development & Code Quality
isort: Automatically sorts Python imports alphabetically and by type. Ensures consistent import ordering across the codebase.
ruff: fast Python linter and formatter. Used for maintaining code quality and enforcing style guidelines.
mypy: for static type checking

Additional Dependencies
matplotlib: Used for data visualization and plotting results
numpy: Provides efficient numerical operations and array manipulations
pandas: Used for data manipulation and analysis of tabular data