# **README for Graph Neural Network Link Prediction**

## **Overview**

This project implements a Graph Neural Network (GNN) using PyTorch Geometric for the task of link prediction in knowledge graphs. The model predicts whether a link (edge) should exist between two nodes in a graph, a common task in areas such as social network analysis and bioinformatics.

To Run: python3 linkPridiction.py

## **Installation Requirements**

* Python 3.x
* PyTorch
* PyTorch Geometric
* Matplotlib (for plotting)
* Pandas (for data handling)
* NetworkX (not used in the current code, but often required for graph operations)

## **Data**

The model is trained and validated on the WN18RR dataset, a popular benchmark in relational learning derived from WordNet. The dataset comprises relationships between words, where each relationship is a type of link between word entities.

## **Code Structure**

### **Modules and Functions**

#### **load\_data(path)**

* **Purpose**: Loads and preprocesses graph data from a file.
* **Input**: Path to the dataset file.
* **Output**: A Data object containing:
  + edge\_index: Tensor representing graph connectivity in COO format (a pair of nodes for each edge).
  + num\_nodes: Total number of unique nodes in the graph.
* **Details**: This function reads a tab-separated file containing source and target nodes, along with their relationship types, converts these into tensor format, and ensures that node indices are consecutive.

#### **Encoder Class**

* **Purpose**: Encodes node features into embeddings using Graph Convolutional Network (GCN) layers.
* **Components**:
  + Embedding layer: Maps each node index to a high-dimensional vector.
  + Two GCN layers: Each layer aggregates features from neighbors to capture local graph topology.
  + Dropout: Regularization technique to prevent overfitting during training.
* **Method**: forward(x, edge\_index)
  + Inputs node features and edge indices.
  + Outputs node embeddings after applying GCNs and activation functions.

#### **LinkPrediction Class**

* **Purpose**: Predicts the existence of links between node pairs using the embeddings generated by the Encoder.
* **Components**:
  + An instance of Encoder.
  + A linear layer (decoder) that projects the node embedding interactions to a prediction score.
* **Method**: forward(edge\_index)
  + Uses only edge indices to fetch node embeddings from the encoder and predict link existence.

#### **train()**

* **Purpose**: Performs a single training epoch, including forward and backward passes.
* **Operations**:
  + Negative sampling: Generates non-existing links to train the model in distinguishing false connections.
  + Forward pass on both positive (actual) and negative (sampled) links.
  + Loss computation and backpropagation.
* **Output**: Returns training loss and accuracy for the epoch.

#### **validate(model, data)**

* **Purpose**: Evaluates the model on the validation dataset.
* **Operations**: Similar to train() but without backpropagation.
* **Output**: Validation accuracy.