Methodology

Step-by-Step Explanation of the Methodology for Link Prediction in Co-authorship Networks

This methodology predicts **future collaborations** between authors in a research network by analyzing their past collaborations (**graph-based features**) and the content of their research papers (**content-based features**). The process consists of **five major steps**:

Step 1: Understanding the Data (Building the Network)

We start with a dataset containing **past co-authorship relationships** and **research paper details**. There are **16249 authors** and **6952 papers.** Following relationships are covered in the dataset:

- **Author–Paper:** Each author is connected to the papers they have written.
- **Paper–Paper:** Papers cite each other.
- **Paper–Journal:** Each paper is associated with a journal.

Step 2: Extracting Features

To predict whether two authors will collaborate in the future, we need useful information about their relationships. We extract two types of features:

Graph-Based Features:
Instead of pre-computed measures (e.g., common neighbors or Jaccard coefficients), we use a Graph Neural Network (GNN) to learn structural representations from the heterogeneous graph.

- **LHGI Module**: The `LHGI` model incorporates two GCNConv layers that process the node features across multiple edge types (extracted via meta-paths).
- **Semantic Attention Layer:** A custom `SemanticAttentionLayer` aggregates outputs from different path views, learning to weight each relation type based on its contribution to the final node embedding. This layer in simple terms deals with the contextual relationship between data

Content-Based Features:
To capture the research content of each author, we assume each author is associated with a set of paper embeddings (pre-computed and stored in `all_data`).

- LSTM Model: The `LstmModel` processes sequences of paper embeddings (each of dimension 1024) using an LSTM network. This is followed by linear transformation and multi-head self-attention to produce a fixed-length embedding that encapsulates the research topics and styles.

Step 3: Combining Features

Fusion of Graph and Content Representations:
For each author in a candidate pair, two representations are computed:

- Graph Embedding: Generated by the LHGI module using the author's neighborhood from the heterogeneous graph.
- Content Embedding: Produced by processing the author's paper embeddings through the LSTM model.

These two embeddings are fused in the `endModel` using an attention-based fusion mechanism (`cat_LstmGnn`). Softmax normalization determines the weight for each representation, allowing the network to adaptively emphasize graph or content features based on context.

Similarity-Based Prediction:
The fused embeddings for both authors in a pair are further transformed and then compared using cosine similarity. The resulting similarity score is used as the prediction signal:

- Higher cosine similarity implies a higher likelihood of future collaboration.
- During training, the model learns to assign high similarity to known collaborative pairs (positive class) and low similarity to non-collaborative pairs (negative class).

Step 4: Training a Machine Learning Model

Loss Function & Optimization:

- We employ the `BCEWithLogitsLoss` (binary cross-entropy with logits) for binary classification.

- The model parameters (both in the **LHGI and LSTM** components as well as fusion layers) are optimized using the Adam optimizer with a learning rate of 0.05.

Training Data & Mini-Batch Sampling:

- The training data is constructed by combining positive and negative author pair samples from provided text files.
- A custom `Data` class loads these pairs, and PyTorch's `DataLoader` is used to generate mini-batches for stochastic training.

Epochs & Training Loop:

We ran the model twice with different epoch numbers (10 and 20 respectively), and during each iteration, the model's output (cosine similarity) is compared with the ground-truth label. Attention weights from the semantic attention layer are logged for further interpretability.

Step 5: Evaluating the Model

To measure how well the model works, we use the following metrics:

Epochs	Accuracy	F1-score	AUC
10	0.7988	0.8303	0.7988
20	0.832	0.8527	0.832