CS725 Assignment 3 Report

GNN-based Retrieval for QA

Professor: Soumen Chakrabarti

Team Members:

Amit Pandey (24M0792)
Sravani Gunnu (24M2116)

April 13, 2025

Contents

1 Introduction **1**

2 Model Architecture **1**

2.1 Overview 1

2.2 Node Representation 1

2.3 Graph Construction 1

2.4 GNN Layers 3

2.5 Query Interaction 3

2.6 Final Projection 4

3 Training Procedure **4**

3.1 Passage-Query Similarity 4

3.2 Binary Cross-Entropy(BCE) 4

3.3 Margin Ranking Loss 5

3.4 Combined Loss 5

3.5 Optimization and Scheduling 5

3.6 Training Configuration 6

4 Evaluation **6**

5 Responses to Assigned Questions **8**

5.1 How is the GNN trained? Supervised or Unsupervised? 8

5.2 What is the best choice of AGGREGATE among MAX, SUM, MEAN, or MIN? 8

5.3 Aggregation Strategy: All Neighbors vs. Selective Sampling 9

5.4 Comparison Against Vanilla Retrieval Methods 10

5.5 What is the effect of number of GNN layers? 10

6 Conclusion **10**

1 Introduction

This project explores Graph Neural Network (GNN)-based retrieval for multi-hop question answering (QA), with the goal of identifying all supporting passages related to a query. Traditional retrieval methods like BM25 and Dense Passage Retrieval (DPR) rely solely on semantic similarity, which often fails to retrieve contextually related but semantically distant passages.

To address this, we construct a Graph of Passages (GoP) using multiple heuristics — including semantic similarity, shared keywords, and sequential adjacency — and apply a GNN to propagate information across the graph. The goal is to improve the relevance ranking of passages, and ultimately pass the top-k passages to a pre-trained language model (T5-small) for final answer generation.

2 Model Architecture

2.1 Overview

We use a 3-layer Graph Attention Network (GATConv) as our GNN model. Each node in the graph represents a passage, and edges capture meaningful relationships between passages. The final node representations are ranked by similarity to the query node to retrieve supporting facts.

2.2 Node Representation

Each passage is embedded using SentenceTransformer (`all-MiniLM-L6-v2`), producing a 384-dimensional vector. These embeddings are then linearly projected to a hidden dimension (256) before being passed into the GNN.

2.3 Graph Construction

The GoP is constructed using the following edge types:

Edge Construction

Edges in the Graph of Passages (GoP) represent different types of relationships between passages. We construct a heterogeneous graph using four edge types:

1. Sequential Edges. These edges connect passages based on their position in the document. If passage i is followed by j, an edge is added with an exponentially decaying weight:

$$w_{i,j}^{\text{seq}} = \exp(-\alpha \cdot \text{distance}_{i,j}), \quad \alpha = 0.5$$

This encourages information flow between adjacent or nearby passages.

2. Keyword Edges. We compute TF-IDF vectors for all passages using `TfidfVectorizer` and calculate pairwise cosine similarity. An edge is added between passages i and j if:

$$\text{cosine}(i, j) > \text{75th percentile}$$

This captures lexical similarity due to shared important terms.

3. Semantic Edges. Semantic similarity is computed using embeddings from `SentenceTransformer`. For each pair of passages (i, j), we calculate:

$$\text{sim}_{i,j} = \cos(\mathbf{h}_i, \mathbf{h}_j)$$

Edges are added if similarity exceeds a dynamic threshold:

$$\text{threshold} = \max(\text{base\_threshold}, \mu + 0.5 \cdot \sigma)$$

where μ and σ are the mean and standard deviation of all pairwise similarities.

If a query is provided, we boost the similarity by:

$$\text{adjusted\_sim} = \text{sim}_{i,j} + 0.2 \cdot \cos(\mathbf{q}, \mathbf{h}_i) \cdot \cos(\mathbf{q}, \mathbf{h}_j)$$

4. Query Edges. A special node `__QUERY__` is introduced to the graph. It is connected to passages whose cosine similarity with the query exceeds 0.4:

$$\cos(\mathbf{q}, \mathbf{h}_i) > 0.4$$

This allows the query to influence the graph propagation process directly.

Summary.

Edge Type	**Criteria**	**Weight**	**Purpose**
Sequential	Passage i to $i+1$ to $i+3$	$e^{-0.5\cdot k}$	Contextual coherence
Keyword	TF-IDF cosine $>$ 75th percentile	Cosine similarity	Lexical relatedness
Semantic	Cosine $>$ dynamic threshold	Cosine + Query Boost	Deeper semantics
Query	Cosine(query, passage) > 0.4	Cosine similarity	Inject query into graph

2.4 GNN Layers

We use 2 layers of GATConv. For a node i at layer l, the update is:

$$\mathbf{h}_i^{(l)} = \text{ReLU}\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} W^{(l)} \mathbf{h}_j^{(l-1)}\right)$$

where $\alpha_{ij}^{(l)}$ is the attention weight between node i and j computed via learned projections. Each layer is followed by:

- Layer normalization
- Residual connection
- Dropout (rate $= 0.3$)

2.5 Query Interaction

If the query node exists in the graph, its representation is compared with all passage nodes using cosine similarity to compute final relevance scores. These scores are used to rank passages.

2.6 Final Projection

The output of the final GNN layer is passed through a linear projection to get the final node embeddings:

$$\mathbf{h}_i^{\text{final}} = W_{\text{out}} \cdot \mathbf{h}_i^{(3)} + b$$

3 Training Procedure

The training objective is to retrieve all relevant passages for a given question by jointly learning to classify relevance and enforce ranking between relevant and irrelevant passages.

3.1 Passage-Query Similarity

After the passage and query nodes are encoded using GNN layers, we compute cosine similarity between the query embedding $\mathbf{q}$ and each passage embedding $\mathbf{p}_i$:

$$\text{sim}_i = \cos(\mathbf{q}, \mathbf{p}_i)$$

To sharpen the similarity distribution, we scale the cosine values by a factor:

$$\text{scaled\_sim}_i = \text{sim\_scale} \cdot \text{sim}_i, \quad \text{sim\_scale} = 20.0$$

This scaling improves the gradient signal for classification and ranking losses.

3.2 Binary Cross-Entropy(BCE)

For each passage i, a binary label $y_i \in \{0, 1\}$ is assigned based on whether it appears in the gold supporting set. We use the Binary Cross-Entropy (BCE) loss to predict relevance:

$$\mathcal{L}_{\text{BCE}} = -\sum_{i=1}^{N} [y_i \log(\sigma(s_i)) + (1 - y_i) \log(1 - \sigma(s_i))]$$

Here, s_i is the scaled cosine similarity and σ is the sigmoid activation.

3.3 Margin Ranking Loss

To ensure that relevant passages are ranked higher than irrelevant ones, we use a pairwise margin ranking loss. For each positive-negative pair, the loss is defined as:

$$\mathcal{L}_{\text{margin}} = \max(0, m - s_{\text{pos}} + s_{\text{neg}}), \quad m = 0.2$$

where s_{pos} and s_{neg} are the similarity scores for relevant and irrelevant passages, respectively. This loss encourages:

$$s_{\text{pos}} > s_{\text{neg}} + m$$

3.4 Combined Loss

The final loss function is a convex combination of the BCE and margin losses:

$$\mathcal{L}_{\text{final}} = \lambda \cdot \mathcal{L}_{\text{margin}} + (1 - \lambda) \cdot \mathcal{L}_{\text{BCE}}, \quad \lambda = 0.5$$

This balances the objective of ranking with absolute relevance classification.

3.5 Optimization and Scheduling

We use the AdamW optimizer with a learning rate of 1×10^{-3} and weight decay 1×10^{-5}. The learning rate is scheduled using Cosine Annealing, which follows the schedule:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t}{T}\pi\right)\right)$$

This schedule gradually lowers the learning rate, improving convergence over epochs.

3.6 Training Configuration

The final training setup is as follows:

- **Epochs:** 50
- **Batch Size:** 32
- **Dropout:** 0.3
- **Cosine Similarity Scale:** 20.0
- **Loss Weight λ:** 0.5
- **Margin:** 0.2
- **Optimizer:** AdamW
- **Scheduler:** CosineAnnealingLR

4 Evaluation

We evaluate our model on the development set using standard retrieval and generation metrics.

Retrieval Metrics

- **F1 Score:** Measures the harmonic mean of precision and recall over the top-5 retrieved passages, comparing them against the gold supporting facts.
- **MRR (Mean Reciprocal Rank):** Evaluates how early the first correct passage is retrieved in the top-k list.

Generation Metric

For answer generation, we use the T5-small model. The top-5 retrieved passages are concatenated with the question and passed as input to T5. The generated output is evaluated using the ROUGE-L metric to measure overlap with the ground truth answer.

Retrieval Results on Dev Set

We compare our GNN-based retriever against sparse (BM25) and dense (DPR) retrieval baselines:

Model	**F1 Score**	**MRR**
BM25 Baseline	0.5111	0.8350
DPR Baseline	0.4524	0.9267
BetterGNN (ours)	**0.5685**	**0.8471**

The GNN retriever achieves the best F1 score, demonstrating improved precision and recall through graph-based reasoning, while maintaining competitive MRR.
The generated answers using T5-small achieved a ROUGE-L score of **0.2295**, indicating a moderate degree of lexical overlap with the ground truth answers.

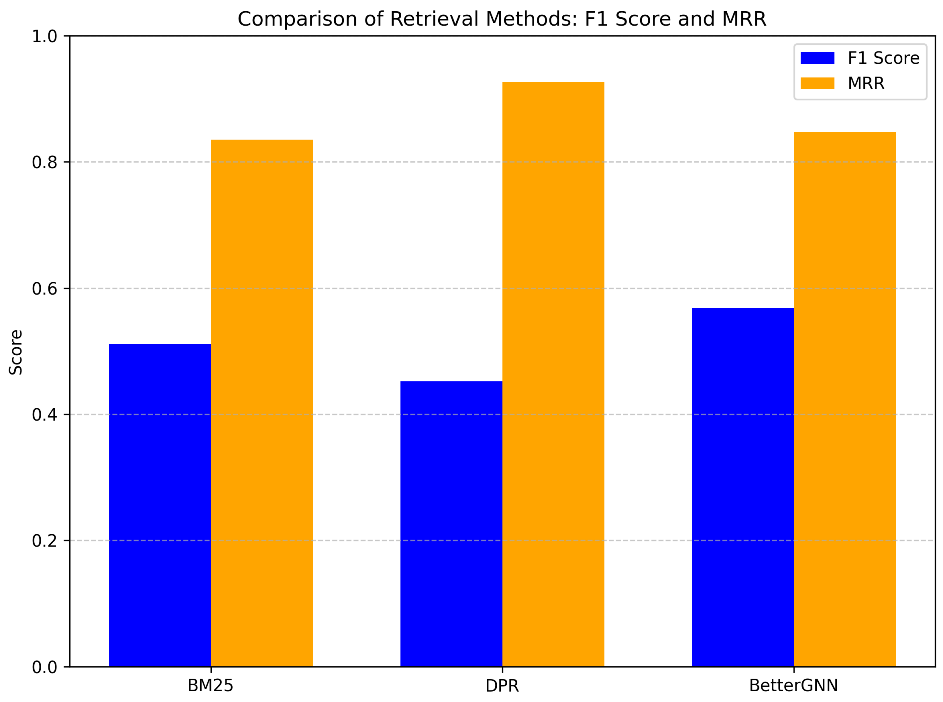

Figure 1: Comparison of Retrieval Methods: F1 Score and MRR

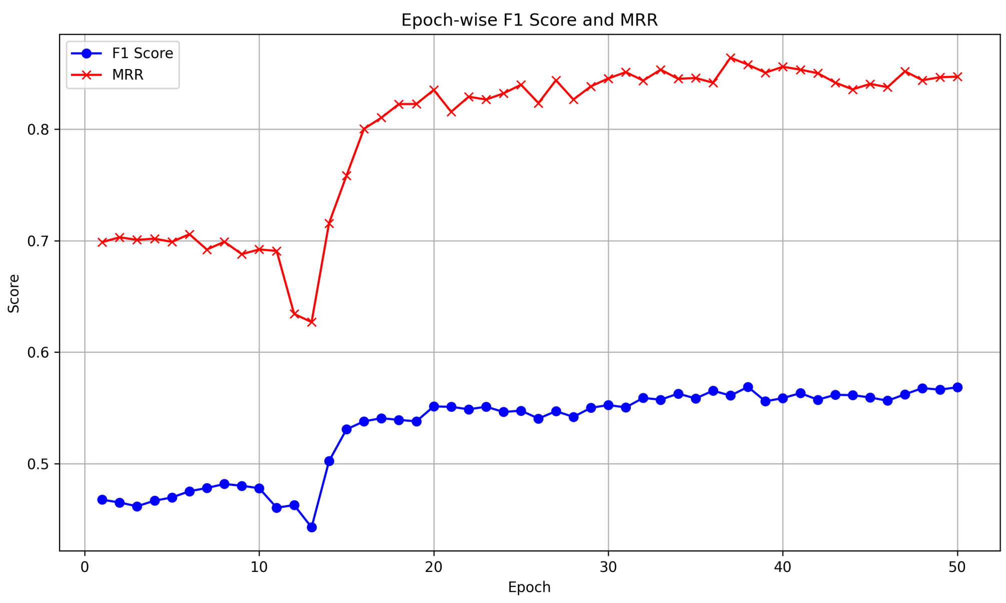

Figure 2: Epoch-wise F1 Score and MRR

5 Responses to Assigned Questions

5.1 How is the GNN trained? Supervised or Unsupervised?

Our GNN is trained using a **supervised learning objective**. Each training instance provides a question, a set of candidate passages, and supporting facts (used as positive labels).

The model is optimized using the Binary Cross-Entropy (BCE) loss applied to cosine similarity scores between the query node and each passage node. This allows the model to learn both the semantic similarity and relevance between the query and passages.

5.2 What is the best choice of AGGREGATE among MAX, SUM, MEAN, or MIN?

We experimentally tested four aggregation functions: `mean`, `max`, `sum`, and `min`, using a 2-layer GNN architecture.

Aggregator Comparison Results:

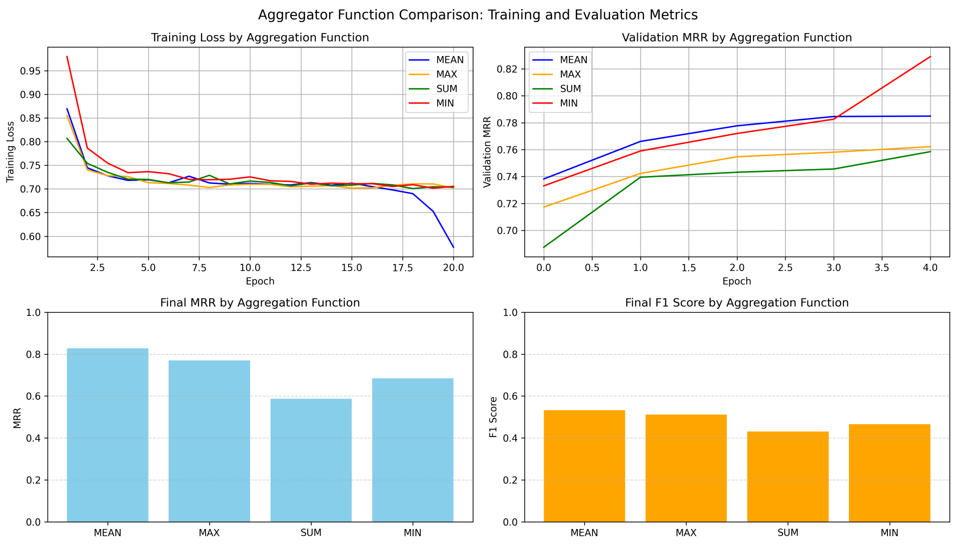

Figure 3: Aggregator Function Comparison: Training and Evaluation Metrics

Aggregator	**F1 Score**	**MRR**
Mean	0.5323	0.8279
Max	0.5116	0.7700
Sum	0.4312	0.5868
Min	0.4659	0.6851

Based on these results, we selected **MEAN** aggregation for all further experiments, as it achieved the best overall performance.

5.3 Aggregation Strategy: All Neighbors vs. Selective Sampling

In our implementation, we aggregate over **all neighbors** of a node in the Graph of Passages. Since the graph is constructed per question and remains of manageable size, full aggregation is computationally feasible and effective.

While no explicit sampling is performed, future work could explore using **attention mechanisms** or importance-weighted sampling to selectively focus on more informative neighbors.

5.4 Comparison Against Vanilla Retrieval Methods

We compare our GNN-based approach with two standard baselines — BM25 (sparse) and DPR (dense semantic):

Baseline Retrieval Results (on 20 dev examples):

Method	F1 Score	MRR
BM25	0.5111	0.8350
DPR	0.4524	0.9267
GNN (2-layer, mean agg.)	**0.5884**	**0.8748**

The GNN model outperforms BM25 and DPR in F1 score by a notable margin. While DPR achieves the highest MRR, GNN provides a stronger balance between early retrieval and overall relevance.

5.5 What is the effect of number of GNN layers?

We studied the effect of GNN depth by varying the number of layers while using mean aggregation. The following results were observed:

Layer Depth Experiment Results:

Number of Layers	F1 Score	MRR
2	**0.5884**	**0.8748**
3	0.4297	0.6468
4	0.4075	0.5849

We observe that using more than two layers **degrades performance**, likely due to **over-smoothing**, where deeper layers lead to indistinguishable node representations. Hence, we chose to proceed with a **2-layer GNN**, which provided the best empirical trade-off between representation power and training stability.

6 Conclusion

In this project, we proposed a Graph Neural Network (GNN)-based retrieval system for multi-hop question answering. By constructing a Graph of Passages (GoP) enriched with semantic, lexical, sequential, and query-based edges, our approach enabled effective message passing and multi-hop reasoning across related passages.

We compared our GNN-based method with strong baselines — BM25 and Dense Passage Retrieval (DPR). Our final model, a 2-layer GNN using mean aggregation, achieved the highest F1 score on the development set, outperforming both baselines in retrieval effectiveness while maintaining a competitive MRR.

We also conducted an in-depth analysis of architectural design choices. Among different aggregation functions, mean aggregation proved most effective. Furthermore, increasing GNN depth beyond two layers led to diminished performance due to over-smoothing, validating our choice of a lightweight, 2-layer configuration.

Overall, our results highlight the value of incorporating structured inter-passage relationships in retrieval tasks. Future work can extend this by exploring attention-based aggregation, end-to-end training with answer generation models, and scaling to larger document collections for open-domain QA.