CS728 Assignment 3

Garima Jain 24M0772

Pratik Shah 24M0856

April 2025

1 Introduction

Modern Question Answering (QA) systems often use Retrieval-Augmented Generation (RAG), where retrievers find relevant passages from a collection of documents to help Large Language Models (LLMs) generate answers. However, traditional retrieval methods treat each passage separately and rank them only based on how similar they are to the query. This approach struggles with complex questions that need multiple related passages, as it ignores the connections and relationships between passages. As a result, passages that are indirectly related to the query but strongly connected to other important passages may be missed, leading to incomplete retrieval of information.

In this assignment, we propose a Graph Neural Network (GNN)-based retrieval framework that models passages as nodes in a graph, with edges representing semantic, contextual, or structural relationships. By propagating relevance signals through these connections, the GNN identifies supporting passages that might be semantically distant from the query but critical to the answer. This method enhances retrieval robustness, ensuring comprehensive context for the LLM and improving answer accuracy. In this assignment, we implement and evaluate this approach, demonstrating how GNNs can bridge the gap between isolated semantic retrieval and holistic, relationship-aware passage ranking.

2 Model Architecture

In this section we discuss the three approaches that were tested during this assignment. This includes two baseline approaches, namely BM25 and DPR, and the proposed GNN approach. We discuss the architecture and implementation of the three models in detail.

2.1 BM25

BM25 (Best Match 25) is a probabilistic information retrieval algorithm that ranks documents based on their relevance to a query. It improves upon traditional TF-IDF by incorporating document length normalization and saturation effects.
The relevance score of a document, D to a query, Q is calculated using the formula:

$$BM25(D,Q) = \Sigma IDF(q_i) \frac{[TF(q_i,D)(k1+1)]}{[TF(q_i,D) + k1(1-b+b\frac{|D|}{avgdl})]}$$

In the assignment for evaluating baseline model, BM25 model is initialized using pre-tokenized corpus and default parameter values are used. Relevance score for all documents is calculated using the formula and sorted to return top-k (in our case, 5) results.

2.2 DPR

Dense Passage Retrieval (DPR) uses a dual-encoder architecture to map questions and passages into a shared dense vector space. The system comprises two main components:

Question Encoder

- Architecture: BERT-base model (bert-base-uncased)
- Input: Natural language question
- Output: 768-dimensional embedding
- Tokenization: Uses DPRQuestionEncoderTokenizer (specialized for questions)
- Pooling: CLS token pooling strategy

Context Encoder

- Architecture: BERT-base model (bert-base-uncased)
- Input: Passage text (title + content)
- Output: 768-dimensional embedding
- Tokenization: Uses DPRContextEncoderTokenizer
- Pooling: CLS token pooling strategy

This implementation follows the RAG (Retrieval-Augmented Generation) paradigm, combining DPR's semantic retrieval with FLAN-T5's generation capabilities. DPR is used for tokenizing generating embeddings and query processing. These queries are then encoded using DPR encoder. The top-k queries are retrieved using FAISS.

2.3 GNN

The Graph Neural Network (GNN) architecture implemented in the code is based on Graph Convolution Networks (GCN).

- Layer Structure:
  - Input Layer: GCNConv($input\_dim->hidden\_dim$)
  - Hidden Layer: GCNConv($hidden\_dim->hidden\_dim$)
  - Scoring Layer: Linear($hidden\_dim->1$)
- Activation: ReLU after first GCN layer
- Aggregation: Mean pooling of neighbor features (default for GCNConv)

The model processes node features (e.g., passage embeddings) and edge indices (graph connectivity) to propagate information across connected nodes. After the first GCNConv layer, ReLU activation introduces non-linearity, followed by a second GCNConv layer to refine node embeddings. The final linear layer outputs relevance scores for each node. A ranking loss function enforces positive passages to score higher than negatives using a margin-based approach. The system handles variable numbers of positive/negative samples and optimizes ranking quality through backpropagation.

3 Graph of Passages

The Graph of Passages is constructed to represent question-answering data as graph structures for training a Graph Neural Network (GNN). Each question-answer pair is processed to extract passages, combining the title and its content. Positive passages containing ground-truth evidence (supporting facts) are identified based on their indices.

Passage embeddings are generated using the `all-mpnet-base-v2` model from Sentence Transformers. Mean pooling with attention mask normalization is applied, followed by L2-normalization to facilitate cosine similarity computation.

Edges between nodes (passages) are constructed using three methods: (1) similarity-based edges, where cosine similarity between embeddings exceeds a threshold of 0.55; (2) explicit edges connecting positive passages to reinforce relationships between supporting facts; and (3) selective neighbor sampling using a k-NN approach (k=10) to retain only the most relevant connections while avoiding self-loops.

The final graph representation consists of passage embeddings as node features, edges formed by combining similarity-based and supporting fact connections, and metadata such as the query, passage titles, and positive passage indices. These graphs are stored as PyTorch Geometric Data objects for training and inference.

4 GNN Training

In this setup, the GNN is trained using a supervised objective. Given a query node and a set of passages, the model is trained to assign higher scores to positive (relevant) passages and lower scores to irrelevant ones. This is commonly implemented via a margin-based ranking loss or binary cross-entropy loss over positive and negative passage nodes.

This approach leverages explicit supervision from positive passage labels, making it more effective for query-passage relevance modeling compared to unsupervised methods like contrastive loss without labeled positives.

4.1 Training Objective

The GNN is trained using a **supervised ranking objective**. The model learns to rank passages based on their relevance to the query by leveraging the ground-truth evidence (positive passages) provided in the dataset. The loss function (`ranking_loss`) ensures that the scores of positive passages are higher than those of negative passages by a margin. Specifically, for each positive passage, the loss penalizes cases where its score is not sufficiently greater than the scores of negative passages. This supervised approach contrasts with unsupervised objectives, which typically rely on clustering or similarity-based heuristics without access to labeled data. By using labeled positive passages, the GNN explicitly learns to distinguish relevant evidence from irrelevant content, making it more effective for retrieval tasks.

Best Choice of Aggregation

For retrieval-focused tasks (e.g., identifying relevant passages), **Max aggregation** is the best choice due to its ability to emphasize dominant signals and prioritize key evidence nodes effectively. For end-to-end QA systems that involve both retrieval and answer generation, **Mean aggregation** may be preferable as it balances contextual richness and semantic preservation.

5 The Choice of Aggregate

The results demonstrate varying performance across different aggregation strategies: Mean, Max, Min, and Sum. Below is a detailed comparison:

- **Mean Aggregation:** Mean aggregation computes the average of neighboring node features, providing a balanced representation by preserving contextual information from all neighbors equally. It performs well in generation metrics (**F1: 0.2339**, **ROUGE-L: 0.2350**), as averaging helps retain semantic richness and smooths out noisy signals. However, its retrieval performance (**MRR: 0.6462**, **F1@5: 0.6258**) is slightly lower than Max aggregation, as it may dilute critical evidence signals.

- **Max Aggregation:** Max aggregation selects the largest feature value from neighboring nodes for each dimension, emphasizing dominant signals in the graph structure. This strategy achieves the best retrieval performance (**MRR: 0.6658**, **F1@5: 0.6319**), as it highlights key evidence nodes that are most relevant for ranking passages. However, it underperforms in generation metrics (**F1: 0.2205**, **ROUGE-L: 0.2218**), likely due to its inability to preserve nuanced semantic information.
- **Min Aggregation:** Min aggregation captures the smallest feature values from neighboring nodes, emphasizing weaker signals in the graph structure. While its retrieval performance (**MRR: 0.6625**, **F1@5: 0.6307**) is competitive with Max aggregation, it slightly outperforms other methods in generation metrics (**F1: 0.2330**, **ROUGE-L: 0.2373**). This suggests that focusing on weaker signals may help retain semantic diversity for answer synthesis.
- **Sum Aggregation:** Sum aggregation computes the sum of neighboring node features, which can lead to magnitude bias as larger neighborhoods dominate smaller ones. It performs worse than other methods in both retrieval (**MRR: 0.6569**, **F1@5: 0.6269**) and generation metrics (**F1: 0.2259**, **ROUGE-L: 0.2264**). This indicates that unnormalized sums are less effective at capturing meaningful relationships between nodes.

Aggregate	**Retrieval MRR**	**Retrieval F1@5**	**Generation F1**	**Generation ROUGE-L**
Mean	0.6462	0.6258	0.2339	0.2350
Max	**0.6658**	**0.6319**	0.2205	0.2218
Min	0.6625	0.6307	0.2330	0.2373
Sum	0.6569	0.6269	0.2259	0.2264

Table 1: Performance Evaluation of GNN training for different aggregates for 2 layers

6 How to Aggregate

Rather than aggregating from all neighbors, our approach uses selective node sampling, which helps in:

- Reducing computation
- Focusing on more relevant or informative nodes
- Avoiding noise from loosely related passages

6.1 Neighborhood Sampling Strategy

We use selective edge sampling, where a fixed number (e.g., 90) of neighbor edges are sampled per node. The sampling is biased toward semantically similar passages (e.g., using prior semantic scores or edge weight distributions), thereby helping the model concentrate on high-signal neighbors.

In this implementation, selective node sampling is employed through the `SelectiveSampler` module to refine edge connections between nodes in the graph:

- **Similarity-Based Sampling:** The sampler uses cosine or Euclidean similarity to prune edges by retaining only the top-10 most similar neighbors for each node while avoiding self-loops. This ensures that only highly relevant connections are preserved in the graph structure.
- **Positive Passage Reinforcement:** Explicit edges are added between all permutations of positive passages (supporting facts) to strengthen relationships between ground-truth evidence nodes.

This hybrid approach reduces noise introduced by irrelevant or weakly connected neighbors while ensuring that supporting facts remain interconnected within the graph structure. The selective sampling strategy improves computational efficiency by limiting edge density and enhances model performance by focusing on meaningful connections.

In this case, selective sampling is crucial because it balances graph complexity with relevance—avoiding overfitting caused by dense graphs while ensuring critical evidence nodes are well-connected for effective ranking and representation learning.

7 Baseline Comparison

The GNN-based retrieval exhibits distinct trade-offs compared to sparse (BM25) and dense (DPR) baselines:

- **Sparse Retrieval (BM25)** excels in MRR (0.8312) and generation metrics (F1: 0.5669), as exact keyword matches provide high-precision results critical for answer synthesis. However, its lower F1@5 (0.4841) reveals limited recall for multi-passage evidence.
- **Dense Retrieval (DPR)** achieves the highest MRR (0.9113) through semantic alignment but underperforms in F1@5 (0.5340) and generation tasks, suggesting latent space matches lack verbatim text needed for LLM responses.
- **GNN Retrieval** bridges this gap with competitive F1@5 (0.6251), leveraging graph structure to identify related passages. However, its lower MRR (0.6518) and generation scores (F1: 0.2347) indicate noise propagation through edges harms ranking precision and answer quality.

Key Insight: GNNs improve evidence recall but struggle with precision-critical tasks. Hybrid approaches (e.g., GNN reranking of BM25/DPR results) could combine the precision of traditional methods with the relational reasoning of graph models.

Model	Retrieval MRR	Retrieval F1@5	Generation F1	Generation ROUGE-L
BM25	0.8312	0.4841	0.5669	0.3340
DPR	0.9113	0.5340	0.3723	0.3926
GNN	0.6518	0.6251	0.2347	0.2353

Table 2: Performance Evaluation for various models

8 Expression vs Compute

The number of GNN layers significantly impacts retrieval performance and computational efficiency. With Mean aggregation, 3 layers achieve the highest F1@5 (0.6485) but lower MRR (0.6245) compared to 2 layers. Four layers show marginal MRR improvements (0.6518) but reduced F1@5 (0.6251), suggesting diminishing returns.

Trade-offs:

- **Expressiveness:** Deeper networks (3-4 layers) capture higher-order neighbor relationships but risk over-smoothing, particularly harmful for MRR which prioritizes top-ranked results.
- **Computational Cost:** Each additional layer increases parameters by 25% (from 2 to 4 layers) while providing limited accuracy gains, making 2-3 layers the cost-effective choice.

The results indicate that shallow architectures (2 layers) balance retrieval accuracy and efficiency, while deeper models may benefit generation tasks through enhanced context propagation at increased computational expense.

Number of Layers	Retrieval MRR	Retrieval F1@5	Generation F1	Generation ROUGE-L
2	0.6462	0.6258	0.2339	0.2350
3	0.6245	0.6485	0.2447	0.2457
4	**0.6518**	**0.6251**	0.2347	0.2353

Table 3: Performance Evaluation for different number of layer with Aggregate = mean

9 Conclusion

The assignment evaluates three retrieval models—BM25, DPR, and GNN—using metrics like Retrieval MRR, Retrieval F1@5, Generation F1, and ROUGE-L. BM25 performs well in precision-critical tasks (MRR: 0.8312, Generation F1: 0.5669) due to its reliance on exact keyword matches but struggles with recall (F1@5: 0.4841) for multi-passage evidence. DPR achieves the highest precision (MRR: 0.9113) by leveraging semantic embeddings but underperforms in generation tasks (F1: 0.3723), as dense embeddings lack verbatim text needed for LLM responses. GNN improves recall (F1@5: 0.6251) by integrating related passages through graph structures but has lower precision (MRR: 0.6518) due to noise propagation through edges.

The choice of aggregation strategies impacts performance:

- Max Aggregation emphasizes dominant signals, achieving the best retrieval performance (MRR: 0.6658, F1@5: 0.6319).
- Mean Aggregation balances retrieval and generation metrics, performing better in generation tasks (F1: 0.2339, ROUGE-L: 0.2350).

The number of GNN layers also affects results:

- 2 layers balance retrieval accuracy and efficiency.
- 3 layers improve recall (F1@5: 0.6485) but reduce precision (MRR: 0.6245).
- 4 layers show diminishing returns with increased computational cost.

10 Conclusion

The GNN-based retrieval framework successfully enhances recall by leveraging relationships between passages but struggles with precision-critical tasks due to noise propagation. BM25 excels in precision but lacks recall for complex queries, while DPR achieves high precision through semantic alignment but is less effective for generation tasks. Hybrid approaches combining GNN reranking with BM25/DPR results could balance precision and recall effectively. The findings highlight that shallow architectures (2 layers) with selective aggregation strategies are cost-effective for real-world QA systems, while deeper models may benefit generation tasks at higher computational expense.