Prompt-Compressed RAG Re-ranking: Balancing nDCG Gain Against Inference Time

Nikhil Reddy **Annabelle Min** **Avi Sharma**

Language Technologies Institute, Carnegie Mellon University
Pittsburgh, PA 15213
{nthambal, annabelm, avisharm}@cs.cmu.edu

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm for enhancing large language models (LLMs) by incorporating external knowledge retrieval. While substantial efforts have focused on improving retrieval quality through advanced re-ranking models such as RankLLaMA, these approaches often introduce significant computational overhead when reranking long documents. In this work, we address this challenge by integrating LLMLingua2, a lightweight prompt compression model, into the RepLLaMA–RankLLaMA pipeline. By selectively compressing input prompts before reranking, our method reduces inference time while preserving critical contextual information, improving the efficiency-accuracy tradeoff in modern RAG systems.

1 Introduction

Retrieval-augmented generation (RAG) systems rely on effective re-ranking practices as part of their overall workflow to ensure the quality and relevance of their response Wang et al. (2024). While retrieval of relevant documents for any query is important, re-ranking is crucial in ensuring that the order of these retrieved documents is refined in order to feed the documents with the most relevant context into large language models (LLMs) first.

Current re-ranking methods prioritize either precision or efficiency Wang et al. (2024). Precision-focused methods use deep language models (DLMs) like RankLLaMA, however, these models are computationally expensive. Efficiency-focused techniques use models like the TILDEv2 model, which achieves low latency in task execution, but trade off fine-grained relevance decisions, especially for more complex queries.

Re-ranking also faces further challenges due to the inherent noise in the retrieved documents. Documents often contain redundant or irrelevant text, making it harder for re-rankers to find subtle meaning differences in the text. This becomes a problem with longer documents, as oftentimes it misses out on relevant text that may help answer the query. These challenges underscore the necessity for reranking methods that more effectively balance accuracy with computational efficiency.

One promising lever for striking this balance is prompt compression, the technique of retaining only the most relevant tokens in a document before passing the document onto the reranker. By compressing passages into shorter, more concise representations, one can reduce the amount of text that the reranker must process. This technique can dramatically reduce the computational cost and latency when reranking.

However, naively compressing documents risks discarding relevant context and tokens that could indicate high relevance for the query at hand. Subtle relevance clues necessary for reranking can be represented in these tokens, but aggressive compression risks losing essential context needed for the reranker to accurately rerank the documents it must process.

In this work, we propose to integrate prompt-level compression as part of the pipeline between RepLLaMA and RankLLaMA. Specifically:

1. **Initial retrieval:** We retrieve an initial set of N documents using RepLLaMA.

2. **Query-guided prompt compression:** We pass these documents thorugh LLMLingua, a prompt compression model that's used to retain a percentage (otherwise known as a **compression rate**) of the original tokens in the document. We explore two methods here: having a constant compression rate and having a dynamic compression rate based on cosine similarity.

3. **Reranking:** We feed the compressed documents into RankLLaMA, a pointwise cross-

encoder model, to rerank the documents according to their relevance to the query.

2 Literature Survey

Current RAG faces challenges in:
1) **low retrieval quality**. In complex and long queries, the model usually chunking the contexts as inputs, which leads to disruptive contextual structure and low retrieval quality. Dong et al. (2023) Besides, low evidence density in long-context documents can also lead to low retrieval quality. In long context, there presents huge noise that impairs the model capacity to correctly identify information needed for query. Chen et al. (2024)
2) **"loss in the middle"**. When retrieving long contexts, the model presents a "lost in the middle" problem, where the retrieval performance degrades dramatically when the relevant information is in the middle of the context window. Liu et al. (2023)
3) **retrieval latency** With the tradeoff between latency, throughput, storage capacity and accuracy, it is still a challenge to ensure state-of-art accuracy benchmark while keeping low latency. RAG introduces significant latency overhead to TTFT latency and end-to-end latency. Shen et al. (2024)

2.1 RankLLaMA

As discussed in Ma et al. (2023), early rerankers were models that were not able to fully capture the semantic relationships between tokens in long, complex documents. In addition, document segmentation often caused a loss of context and introduced noise. This would impair the model when it tries to identify relevant content for a query.

In the paper from Ma et al. (2023), researchers tried to explore the capabilities of LLMs like LLaMA-2 in multistage retrieval pipelines. They utilized RankLLaMA, a LLaMA-2 used as pointwise rerankers, training on constrastive loss with hard negatives, with each batch containing a positive document and several hard negatives per query. The researchers also used LoRA for memory-efficient fine-tuning.

This model, RankLLaMA, outperformed other reranking baselines, improving by 3-5% on relevance scores over other reranking baselines on passage ranking for passage ranking datasets such as MS MARCO. It's also able to process entire long documents, preserving contextual integrity and improving retrieval quality.

2.2 Related Work

We explored different models that overcome these challenges separately: RankLLaMA Ma et al. (2023), LongRAG Zhao et al. (2024), RankRAG Yu et al. (2024), CoRAG Wang et al. (2025), CRAG Yan et al. (2024).

LongRAG was an alternative we explored for this project. Zhao et al. (2024). It was designed for long-context QA in order to address issues such as "lost in the middle" and redundancy. LongRAG improves initial retrieval with sliding window expansion in order to preserve semantic continuity. It also uses hybrid retrievers (dual encoder and cross encoder), LLM-augmented extractor, and a Chain-of-Thought (CoT) filter to recover and reason over broader semantic structure of documents before answer generation. And while it achieves a 17% accuracy boost over vanilla RAG systems, it is heavily dependent on the first retrieval and prompt engineering. For these reasons, we opted against utilizing LongRAG.

RankRAG was introduced as a way to optimize both retrieval ranking and answer generation in the same LLM. Yu et al. (2024). It first is fine-tuned on instruction-following datasets (QA, conversations, etc). It then is further fine-tuned on QA and context-ranking tasks. RankRAG outperformed baselines such as GPT-4 on multi-hop QA tasks. However, because of poor modularity, we wouldn't be able to distinguish the reranking behavior from generation, which is why we decide not to apply RankRAG.

Chain-of-Retrieval-Augmented Generation, or CoRAG, was introduced as a model to generate retrieval sub-queries and sub-answers. Wang et al. (2025). For this paper, existing QA datasets are augmented to automatically generate intermediate sub-queries and sub-answers. The model was fine-tuned on predicting the next sub-query, intermediate sub-answer, and final answer. And for inference, the model applied iterative retrieval and reasoning. This model is considered to be the state of the art for multi-hop QA benchmarks and can generalize across domains. However, because of its high computational cost due to repeated retrieval and decoding during inference, we chose against CoRAG for our project.

Finally, we looked into Corrective Retrieval-Augmented Generation (CRAG), as it utilizes a lightweight retreival evaluator that judges each retrieved document before answering the user's query. Yan et al. (2024). The evaluator here is a fine-tuned

T5 model that assigns a confidence level to the retrieval. Based on this confidence level, the retrieved document is classified into one of three actions: Correct, Incorrect, or Ambiguous. If the retrieved document is Correct, CRAG decomposes the retrieved documents into smaller segments, filters for relevance, and recomposes a high-quality input. If the retrieved document is incorrect, CRAG discards the retrieved results and performs a large-scale web search. If the retrieved document is ambiguous, both the refined internal docs and external web results are combined to inform the final answer. Through CRAG, researchers were able to see a significant increase in performance when evaluating CRAG on biography QA tasks, and noted its robustness against poor retrievals. However, we ultimately chose against this due to the heavy architecture overhead, as well as making the system more fragile for a simple reranking study.

3 Baseline Models

3.1 RepLLaMA

RepLLaMA is a fine-tuned adaptation of LLaMA-2 Ma et al. (2023). It uses a dense retriever designed to enhance first-stage retrieval quality in RAG systems. Using a bi-encoder architecture, it independently encodes queries and documents, and computes relevance using a dot product of the dense vectors.

It surpasses other dense retrievers (i.e. bi-SimLM) in standard benchmarks on datasets such as the MS MARCO datases. Also, due to LLM pretraining on long contexts, it handles long context windows better without needing segmentation.

Despite its effectiveness, RepLLaMA incurs high computational and memory costs during both training and inference. However, for our purposes, since we are just doing inference on our datasets, we leverage RepLLaMA for its ability to improve the quality of retrieval in a RAG system.

3.2 LLMLingua

For prompt compression between our retrieval and reranking steps, we utilize a model called LLMLingua (Jiang et al., 2023). This model compresses prompts without compromising semantic integrity. It protects crucial sections like instructions and questions from aggressive compression while applying stronger compression to redundant demonstrations. This component-aware strategy preserves the logical structure and intent of the original prompt.

LLMLingua compresses tokens iteratively, modeling the conditional dependencies between tokens in order to ensure that the system maintains the semantic coherence even with high compression rates being passed in. By doing this, LLMLingua can retain critical reasoning steps and semantic structures.

Through various experimentation, compressed prompts from LLMLingua can preserve semantic content, even while drastically reducing input length. From this, we can deduct that LLMLingua offers a lightweight and practical solution for increasing the speed of inference for tasks such as reranking.

3.3 RankLLaMA

RankLLaMA is an adaptation of LLaMA-2 Ma et al. (2023). It's been fine-tuned as a pointwise cross encoder. It takes in a whole query and candidate document into a single LLaMA decoder block. By it's projection of the final hidden state through a linear layer, it can obtain a relevance score per query-document pair.

On query-document datasets such as MS MARCO, DL19 and DL20 datasets, RankLLaMA achieves a strong reranking performance, as mentioned in the previous section. However, it requires substantial GPU resources, requiring 16*32G V100 GPUs for training and therefore making it too costly for many researchers.

By introducing LLMLingua before passing retrieved documents into RankLLaMA for reranking, we hope to reduce the latency and computational overhead of reranking the documents.

4 Methodology

Our baseline models (RepLLaMA+ RankLLaMA) can achieve high relevance performance, but at cost of high latency, memory consumption and high inference cost, especially when reranking long document contexts. Also, simply scaling up model size from 7B to 13B gives diminishing returns in accuracy. Thus,our model pipeline aims to balance relevance performance and inference efficiency in reranking for RAG systems by integrating prompt compression into the reranking process.

4.1 Model Description

As shown in Figure 1, we use a three-stage retrieval, compress and reranking pipeline. 1) The retriever is

a pre-trained RepLLaMA bi-encoder that retrieves the top-200 passages for each query. 2) These passages are then compressed using LLMLingua2, a Transformer encoder-based prompt compressor. 3) The compressed query-passage pairs are subsequently reranked by RankLLaMA, a cross-encoder model that predicts relevance scores for each pair. LLMLingua2 reduces input token lengths while preserving semantic information, thus accelerating reranking and reducing memory costs without significant performance loss.

4.2 Dataset Description

For our project, we evaluated the performance of the 7B LLaMA-based re-ranker on the TREC Deep Learning Tracks for 2019 (DL19) and 2020 (DL20). These datasets are widely used benchmarks in information retrieval, focusing on passage and document ranking tasks.

TREC DL19 Dataset (Craswell et al., 2020)
-Corpus: A collection of 3.2 million documents, each containing a title, body text, and metadata. The corpus is shared between passage and document ranking tasks.
-Queries: The dataset contains 43 test queries with human-graded relevance judgments. Queries are natural language questions or statements.
-Relevance Judgments: Documents and passages are judged on a four-point scale: Not Relevant (0), Relevant (1), Highly Relevant (2), and Perfect (3) For binary relevance metrics like nDCG@10, levels 1–3 are considered relevant

TREC DL20 Dataset (Craswell et al., 2021)
The TREC DL20 dataset builds on the DL19 dataset with additional queries and updated relevance judgments.
-Corpus: It uses the same set of 3.2 million documents as the DL19 dataset.
-Queries: The dataset includes 54 test queries with graded relevance annotations.
-Relevance Judgments: The same four-point scale is used as in DL19, ensuring consistency in evaluation metrics.

Significance: These datasets are critical benchmarks for testing retrieval models in both academic and industrial contexts. They provide a standardized way to compare the effectiveness of retrieval systems under realistic conditions, including sparse and dense retrieval settings.

4.3 Experimental Setup

Platform and Resources
The experiments were conducted on Google Colab Pro, utilizing an NVIDIA A100 GPU with 40GB of VRAM. This provided sufficient computational resources for running the 7B LLaMA-based re-ranker model. However, attempts to use the larger 13B version of the model resulted in out-of-memory errors due to the high GPU memory requirements.

4.4 Static Compression Rate Methodology

In our model pipeline, we first apply fixed-rate compression strategy to compress retrieved documents before reranking them. After retrieving the top-200 passages per query using RepLLaMA, we concatenate the query and each passage into a single prompt. We then apply LLMLingua2, a transformer-based compressor, to reduce the token length of the prompt. The compression rate is a hyperparameter that determines the fraction of tokens to retain for all inputs uniformly. For instance, a compression rate of 0.3 remains approximately 30% of the original prompt tokens. We tried compression rate from 0.1 to 0.9.

LLMLingua2 operates by framing compression as a token classification task: for each token, the model predicts whether it should be retained or dropped. We will discuss its loss function later. This decision is made based on the semantic importance of the token within the full bidirectional context of the prompt. After compression, the shortened prompts are fed into RankLLaMA, a reranker model that computes a relevance score by contrastive loss for each document-query pair.

The pipeline aims to strike a balance between preserving ranking quality (measured by nDCG@5,nDCG@10, nDCG@20) and reducing end-to-end inference time.

4.5 Dynamic Compression Rate Methodology

We also tried dynamic compression rate in prompt compression step, unlike fixed compression where all documents are compressed uniformly (e.g. always 50% tokens kept), in dynamic compression, the compression rate is adaptive based on individual document-query pairs. The goal is to compress

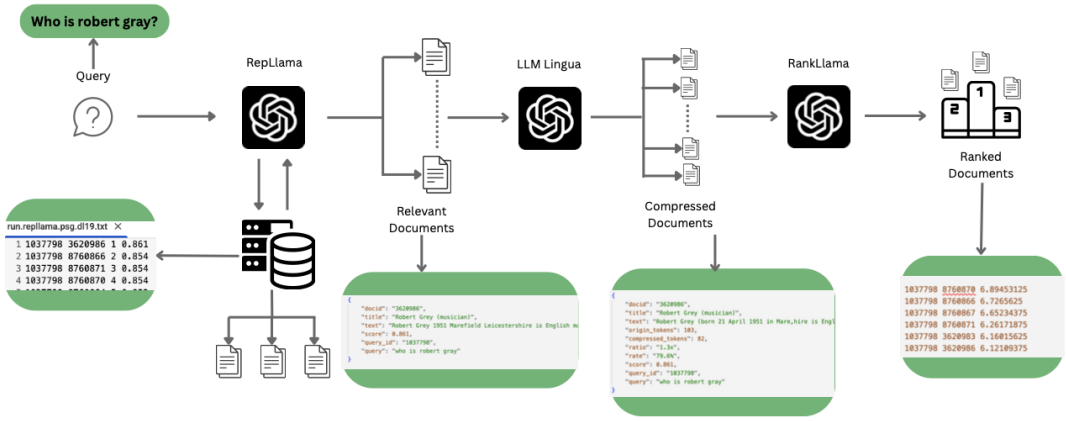

Figure 1: Overview of our RAG reranking pipeline: RepLLaMA retrieves top-200 passages, LLMLingua compresses the prompts, and RankLLaMA reranks the compressed passages.

aggressively for less important documents and retain more for important documents.

To enable dynamic prompt compression, we first compute semantic similarity scores between the query and each retrieved passage chunk. We use the all-MiniLM-L6-v2 model, a lightweight transformer-based encoder from Sentence-Transformers, to map both the query and passage chunks into a shared embedding space. Wang et al. (2020) Cosine similarity is then computed between the query embedding and each chunk embedding, defined as the normalized dot product of the two vectors.

High cosine similarity indicates strong semantic alignment between the query and the passage chunk, while lower values suggest weaker relevance. These similarity scores serve as a proxy for chunk importance, informing our adaptive compression strategy. The distribution of cosine similarity between query and passage is shown in Figure 2. We observe that the majority of cosine similarity scores fall between 0.3 and 0.7 for both the DL19 and DL20 datasets, with a concentration around 0.4–0.6. This distribution suggests that a dynamic compression policy based on similarity thresholds is well-justified, as it allows us to differentially compress less relevant chunks while preserving more information for moderately to highly relevant chunks. Notably, the overlap between DL19 and DL20 distributions indicates that the dynamic compression strategy generalizes across datasets.

To implement dynamic compression, we define two cosine similarity thresholds: a lower threshold at 0.4 and an upper threshold at 0.6. If a passage chunk's similarity with the query falls below the lower threshold, it is treated as less relevant and compressed more aggressively, retaining fewer tokens. Conversely, if a passage chunk's similarity exceeds the upper threshold, it is considered highly relevant and compressed minimally to preserve semantic richness. For chunks with similarity between the two thresholds, we set a fixed compression rate as a hyperparameter. This threshold-based policy enables finer control over the compression strength applied to different passage chunks, helping prioritize the retention of critical information. By dynamically adjusting the compression rate rather than applying a uniform fixed rate, we aim to reduce the end-to-end latency of the reranking stage without sacrificing relevance for the most important content. This approach better adapts to the semantic variability across retrieved docu-

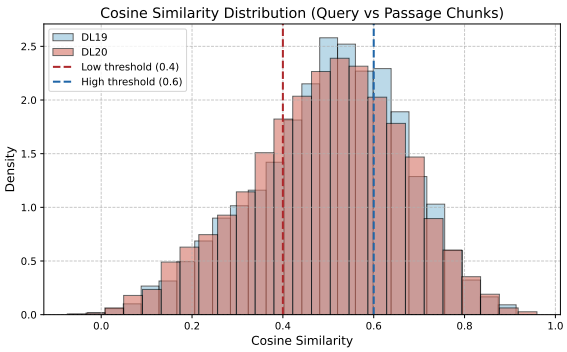

Figure 2: Distribution of cosine similarities between queries and passage chunks across DL19 and DL20 datasets. Red and blue dashed lines indicate dynamic compression thresholds at 0.4 and 0.6 respectively.

ments and addresses the inefficiencies inherent in static compression methods. All the thresholds and compressed rate are hyperparameters and we will discuss the results in section five.

4.6 Evaluation Methodology

We evaluate the quality of the reranked results using Normalized Discounted Cumulative Gain (nDCG) at ranks 5, 10, and 20, denoted as nDCG@5, nDCG@10, and nDCG@20. nDCG is a standard evaluation metric in information retrieval that accounts for both the graded relevance of documents and their ranking positions.

First, the Discounted Cumulative Gain (DCG) at rank k is computed as:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

where rel_i is the graded relevance label of the document at rank i.

The numerator $2^{rel_i} - 1$ gives exponentially higher rewards to highly relevant documents, while the denominator $\log_2(i+1)$ applies a logarithmic penalty to lower-ranked documents, emphasizing the importance of retrieving relevant documents early.

To normalize across queries with varying numbers of relevant documents, the Ideal DCG (IDCG) is calculated by sorting the documents in the ideal order:

$$IDCG@k = \sum_{i=1}^{k} \frac{2^{rel_i^*} - 1}{\log_2(i+1)}$$

where rel_i^* denotes the relevance of the document at position i in the ideal ranking.

The Normalized DCG (nDCG) is then computed as:

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

yielding a score between 0 and 1, where 1 represents a perfect ranking.

We use the graded relevance labels provided in the TREC DL19 and DL20 datasets, where document relevance is annotated on a 0–3 scale. Levels 1–3 are considered relevant for binary settings but are preserved as graded scores for nDCG calculation.

By reporting nDCG@5, nDCG@10, and nDCG@20, we capture both early precision (important for practical retrieval systems) and overall ranking quality across longer retrieved lists. This comprehensive evaluation ensures that our compression strategies improve retrieval effectiveness without sacrificing relevance at critical ranks.

5 Results

5.1 Baseline Results

Dataset	ndgc@5	ndgc@10	ndgc@20
DL19	78.09	76.53	74.53
DL20	76.58	75.12	72.46

Table 1: Results

5.2 Static Compression Results

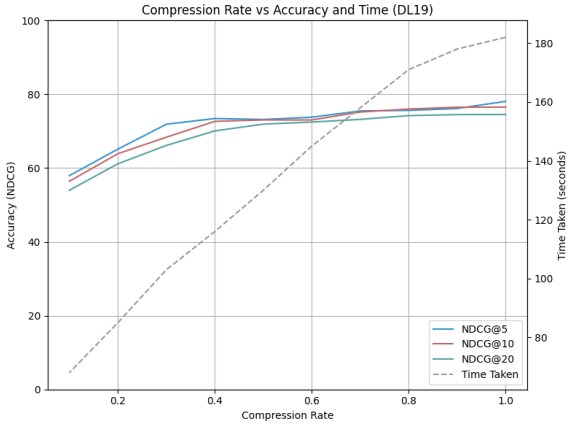

Figure 3: Compression Rate vs Accuracy and Time for DL19 Dataset

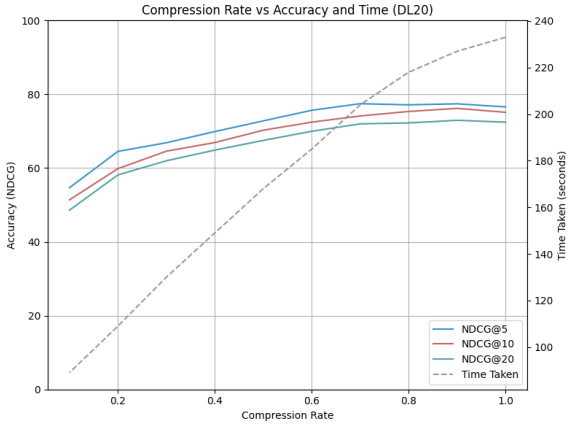

Figure 4: Compression Rate vs Accuracy and Time for DL20 Dataset

5.3 Dynamic Compression Results

We select top 5 best performing nDCG@10 ablations for DL19 and DL20 datasets.

Upper Bound	Lower Bound	High Rate	Mid Rate	Low Rate	Time (s)	nDCG@5	nDCG@10	nDCG@20
1.0	1.0	1.0	1.0	1.0	182	**78.09**	**76.53**	**74.53**
0.55	0.45	0.8	0.5	0.3	148	76.94	76.29	74.82
0.6	0.4	0.8	0.6	0.2	148	76.67	76.02	74.39
0.55	0.45	1.0	0.5	0.3	162	77.58	75.99	74.43
0.6	0.4	0.8	0.5	0.3	142	76.63	75.84	74.36
0.6	0.4	0.8	0.6	0.3	150	76.55	75.82	74.42

Table 2: Top 5 dynamic compression results on DL19

Upper Bound	Lower Bound	High Rate	Mid Rate	Low Rate	Time (s)	nDCG@5	nDCG@10	nDCG@20
1.0	1.0	1.0	1.0	1.0	233	76.58	75.12	72.46
0.55	0.45	0.9	0.5	0.3	192	**78.13**	**76.34**	**72.68**
0.55	0.45	0.8	0.5	0.3	184	77.32	75.56	72.59
0.6	0.4	0.9	0.5	0.3	186	77.20	75.50	71.98
0.6	0.4	0.8	0.6	0.3	185	76.06	75.26	71.71
0.6	0.4	0.8	0.6	0.2	184	76.06	75.26	71.81

Table 3: Top 5 dynamic compression results on DL20

6 Results Analysis

Our experimental results demonstrate a clear trade-off between compression rate, ranking performance, and inference time in RAG systems. For static compression, we observe that:

- Compression rates between 0.4-0.6 offer an optimal balance, maintaining 95%+ of baseline nDCG while reducing inference time by 15-25%
- Performance degradation becomes more pronounced at compression rates below 0.3, particularly affecting nDCG@20 metrics
- The DL19 dataset shows more resilience to aggressive compression compared to DL20

For dynamic compression, our similarity-based approach produces several key insights:

- The best configuration (upper bound: 0.55, lower bound: 0.45) on DL20 achieved an nDCG@5 of 78.13, surpassing the baseline's 76.58 by 1.55 points while reducing inference time by 17.6%
- Documents with low query relevance benefit from aggressive compression (0.3 rate), removing noise that might confuse the reranker
- Documents with high relevance preserve performance when compressed minimally (0.8-0.9 rate)
- The mid-relevance compression rate (0.5) balances token retention with computational efficiency

The dynamic strategy demonstrates that content-aware compression not only improves efficiency but can enhance ranking quality by filtering irrelevant content before the reranking stage.

7 Limitation

Our approach shows promising results, but a few limitations are:

- Our evaluation is limited to the TREC DL19 and DL20 datasets, which may not fully represent all real-world retrieval scenarios and document types.
- Hardware constraints prevented testing larger models (13B version) due to memory limitations on our A100 GPU with 40GB VRAM, indicating potential scalability challenges with larger models.
- The dynamic compression approach introduces additional hyperparameters (similarity thresholds, compression rates for different relevance levels) that require careful tuning.

8 Conclusion

Our work addresses a significant challenge in RAG systems which is the computational overhead of reranking long documents while maintaining retrieval quality. By selectively compressing less relevant content while preserving critical information, we've created an approach that improves the efficiency-accuracy tradeoff in modern RAG pipelines.

9 Assignment Notes

1) Our project satisfies requirement (1) from the assignment description by introducing new techniques for an existing task with significant technical sophistication. We've developed an innovative approach to improve Retrieval-Augmented Generation (RAG) by integrating prompt compression between retrieval and reranking stages. Specifically, we implement LLMLingua2 as a lightweight prompt compressor to optimize the flow of information from RepLLaMA (retriever) to RankLLaMA (reranker). The technical sophistication of our approach is demonstrated through:

- Implementation of both static and dynamic compression strategies
- Development of a semantic similarity-based thresholding mechanism that adaptively compresses documents based on their relevance to queries
- Careful balancing of nDCG performance against inference time reduction
- Comprehensive evaluation across multiple compression rates and parameter settings

2) There is meaningful overlap between our HW3 and HW4, with HW4 representing a substantial advancement and implementation of the ideas proposed in HW3. HW4 represents the culmination of our project, moving from the theoretical understanding and error analysis in HW3 to a concrete implementation with empirical results demonstrating the viability of our approach. Key advancements in HW4 include:

- Full implementation of static compression with variable compression rates
- Development of a dynamic compression strategy using cosine similarity thresholds
- Comprehensive experimental evaluation showing the trade-offs between compression rates, inference time, and ranking quality (nDCG metrics)
- Detailed analysis of how different compression strategies impact performance across DL19 and DL20 datasets
- Visualization and quantification of the efficiency-accuracy trade-off

10 Appendix

Upper Bound	Lower Bound	High Rate	Mid Rate	Low Rate	Time Taken (s)	NDCG@5 **DL19**	NDCG@10 **DL19**	NDCG@20 **DL19**
1.0	1.0	1.0	1.0	1.0	182	78.09	76.53	74.53
0.6	0.4	1.0	0.5	0.3	155	76.62	75.42	74.12
0.6	0.4	0.9	0.5	0.3	150	74.92	74.37	73.86
0.6	0.4	0.8	0.5	0.3	142	76.63	75.84	74.36
0.6	0.4	0.7	0.5	0.3	134	75.38	74.68	73.69
0.6	0.4	0.6	0.5	0.3	128	74.51	73.1	72.59
0.6	0.4	0.5	0.5	0.3	123	72.41	72.09	71.07
0.6	0.4	1.0	0.7	0.2	163	76.32	75.34	73.72
0.6	0.4	1.0	0.5	0.2	156	76.62	75.48	74.22
0.6	0.4	1.0	0.5	0.1	156	76.39	75.2	73.87
0.6	0.4	0.8	0.6	0.3	150	76.55	75.8	74.42
0.6	0.4	0.8	0.7	0.3	155	76.21	75.79	73.91
0.6	0.4	0.8	0.6	0.2	148	76.67	6.02	74.39
0.55	0.45	1.0	0.5	0.3	162	77.58	75.99	74.43
0.55	0.45	0.9	0.5	0.3	155	75.97	75.45	74.25
0.55	0.45	0.8	0.5	0.3	148	76.94	76.29	74.82
0.55	0.45	0.7	0.5	0.3	137	75.54	74.75	74.06
0.55	0.45	0.6	0.5	0.3	129	74.74	73.24	73.05

Table 4: Dynamic Compression with DL19

Upper Bound	Lower Bound	High Rate	Mid Rate	Low Rate	Time Taken (s)	NDCG@5 **DL20**	NDCG@10 **DL20**	NDCG@20 **DL20**
1.0	1.0	1.0	1.0	1.0	233	76.58	75.12	72.46
0.6	0.4	1.0	0.5	0.3	191	76.15	75.25	72.04
0.6	0.4	0.9	0.5	0.3	186	77.2	75.5	71.98
0.6	0.4	0.8	0.5	0.3	180	76.59	74.88	71.74
0.6	0.4	0.7	0.5	0.3	170	76.32	74.61	71.04
0.6	0.4	0.6	0.5	0.3	162	76.05	72.82	70.33
0.6	0.4	0.5	0.5	0.3	156	72.25	70.64	67.35
0.6	0.4	1.0	0.7	0.2	202	76.45	75.21	67.35
0.6	0.4	1.0	0.5	0.2	190	76.15	75.16	71.75
0.6	0.4	1.0	0.5	0.1	188	76.15	74.96	71.6
0.6	0.4	0.8	0.6	0.3	185	76.06	75.26	71.71
0.6	0.4	0.8	0.7	0.3	193	76.55	75.08	72.08
0.6	0.4	0.8	0.6	0.2	184	76.06	75.26	71.81
0.55	0.45	1.0	0.5	0.3	199	76.1	74.98	72.17
0.55	0.45	0.9	0.5	0.3	192	78.13	76.34	72.68
0.55	0.45	0.8	0.5	0.3	184	77.32	75.56	72.59
0.55	0.45	0.7	0.5	0.3	172	75.97	74.59	71.62
0.55	0.45	0.6	0.5	0.3	162	75.73	72.62	69.99

Table 5: Dynamic Compression with DL20

References

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17754–17762.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. *Preprint*, arXiv:2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the trec 2019 deep learning track. *Preprint*, arXiv:2003.07820.

Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. 2023. A survey on long text modeling with transformers. *Preprint*, arXiv:2302.14502.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *Preprint*, arXiv:2310.05736.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-tuning llama for multi-stage text retrieval. *Preprint*, arXiv:2310.08319.

Michael Shen, Muhammad Umar, Kiwan Maeng, G. Edward Suh, and Udit Gupta. 2024. Towards understanding systems trade-offs in retrieval-augmented generation model inference. *Preprint*, arXiv:2412.11854.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. 2025. Chain-of-retrieval augmented generation. *Preprint*, arXiv:2501.14342.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Preprint*, arXiv:2002.10957.

Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, Ruicheng Yin, Changze Lv, Xiaoqing Zheng, and Xuanjing Huang. 2024. Searching for best practices in retrieval-augmented generation. *Preprint*, arXiv:2407.01219.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *Preprint*, arXiv:2401.15884.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Preprint*, arXiv:2407.02485.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. 2024. Longrag: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. *Preprint*, arXiv:2410.18050.