Characterizing Prompt Compression Methods for Long Context Inference

Siddharth Jha [1] **Lutfi Eren Erdogan** [1] **Sehoon Kim** [1] **Kurt Keutzer** [1] **Amir Gholami** [1]

Abstract

Retrieval-augmented generation has become a popular paradigm to integrate custom data sources with large language models (LLMs). However, this often leads to large contexts of tens of thousands of tokens. Long context inference presents challenges at the system level with increased compute and memory requirements, as well as from an accuracy perspective in being able to reason over long contexts. This has led to prompt compression techniques that aim to reduce the size of provided context, while preserving key information. However, despite the wide variety of recently proposed methodologies for compressing long contexts, little standardized analysis has been done to analyze the behavior of different methods across compression rates and tasks. In this paper, we provide a comprehensive characterization and evaluation of prompt compression methods, giving insight into building compression techniques for long context applications. We analyze extractive compression, summarization-based abstractive compression, and token pruning methods. We find that extractive compression is a strong choice, often being able to compress over $10\times$ with minimal accuracy loss. Token pruning demonstrates marginal improvements over extractive compression on summarization tasks. Furthermore, the performance of abstractive compression can be significantly enhanced, by up to 10 points in multi-document QA tasks at $30\times$ compression, through the generation of query-aware summaries.

1. Introduction

In recent years, the use of LLMs has experienced exponential growth, leading to a surge in applications that manage extensive textual contexts. The ability to perform long context inference is crucial in fields like legal or financial document analysis (Wu et al., 2023; Yang et al., 2023b), summarization (Xiao & Carenini, 2019), and interactive systems maintaining ongoing dialogues (Packer et al., 2023). However, building applications that support long prompts presents significant system-level challenges, including increased computational demands, memory requirements, and costs (Hooper et al., 2024; Kim et al., 2023). There is also the potential for a decline in the model's reasoning capabilities over extended sequences (Liu et al., 2024). Consequently, numerous prompt compression methods have been proposed, which aim to condense prompt lengths while preserving essential information. Despite growing interest in prompt compression techniques, little is known about the behavior of such techniques due to a lack of standardized analysis, making it challenging for practitioners to choose the appropriate method for different applications. For example, certain methods evaluate on context sizes of tens of thousands of tokens, while others on only a few hundred. Apart from initial context length, the evaluated compression rates and tasks also greatly vary. This paper addresses these challenges by presenting a detailed characterization and evaluation of prompt compression methods, which are vital for optimizing the efficiency and effectiveness of applications that rely on long context LLM inference. In particular, we make the following contributions:

- In Section 2, we characterize methods into extractive compression, abstractive compression, or token pruning. We further distinguish methods as being query-agnostic or query-aware.
- In Section 3, we evaluate each paradigm on three single-document QA, multi-document QA, and summarization datasets. In Section 4, we study the impact of chunk size and query-aware abstractive summarization.
- Our findings reveal that extractive compression is a strong choice, often being able to compress over $10\times$ with minimal accuracy loss. Token pruning demonstrates marginal improvements over extractive compression on summarization tasks. Furthermore, the performance of abstractive compression can be significantly enhanced, by up to 10 points in multi-document QA tasks at $30\times$ compression, through the generation of query-aware summaries.

[1]UC Berkeley. Correspondence to: Siddharth Jha <sidjha@berkeley.edu>.

Presented in Efficient Systems for Foundation Models workshop in the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

2. Prompt Compression Background

Prompt compression is the process of taking a long prompt and distilling only the necessary information to solve the task. This can be done by either directly manipulating the text or by manipulating text embeddings. As an example of the latter, (Tan et al., 2024) uses an encoder model to produce compressed token embeddings from the original context, which is inputted to a fine-tuned decoder model. While embedding-based compression methods show strong compression performance, such methods require extensive fine-tuning and significant changes to the inference pipeline.

Therefore, our main focus is on direct text manipulation as it requires minimal changes to the inference pipeline and can be used with both open-source models and proprietary LLM API providers. Text-based prompt compression methods fall into three categories: token pruning, abstractive compression, and extractive compression. This section provides an overview of each approach, illustrated in Figure 3.

2.1. Token Pruning Based Compression

Token pruning methods perform compression by discarding irrelevant tokens. Selective-Context (Li et al., 2023) uses a small language model to judge self-information of tokens. Then, tokens with low self-information are pruned from the original prompt. LLMLingua (Jiang et al., 2023b) is a similar method to Selective-Context but uses perplexity to determine the importance of tokens. LLMLingua first performs coarse-grained pruning by removing entire in-context examples and then performs fine-grained token pruning on the prompt. LongLLMLingua (Jiang et al., 2023c) is a modification of LLMLingua designed for long context prompt compression. Unlike LLMLingua, LongLLMLingua considers the perplexity of the question when conditioned on supporting documents to determine which documents are most relevant. After performing coarse-grained compression by removing irrelevant documents, fine-grained token pruning is performed by considering the perplexity of tokens before and after being conditioned on the question. The drop in perplexity after conditioning on the question is used to judge the relevance of a token. Tokens with low relevance are pruned. There has also been extensive research on token pruning methods for white-box Transformer models (Goyal et al., 2020; Kim & Cho, 2020; Kim et al., 2022; Wang et al., 2021). Such methods utilize the Transformer model's attention map at each layer in order to determine which tokens are least attended to by other tokens. These tokens are pruned before the sequence proceeds to the next layer in the Transformer. For the purposes of black-box prompt compression, a smaller white-box model may be used for token pruning, with the unpruned tokens from the white-box model being sent to the black-box LLM.

2.2. Abstractive Compression

Abstractive compression techniques rely on summarization techniques to reduce the length of the original context. RECOMP's (Xu et al., 2023) abstractive compressor is a fine-tuned T5-Large (775M) model (Raffel et al., 2020) that summarizes the initial context into a more compact form. By prompting the summarizer with the question at inference time, they generate query-aware summaries. In the fine-tuning training data, they drive the summarization model to produce an empty string if a summarized context leads to performance degradation on the downstream task. To omit the fine-tuning process in RECOMP, it is also possible to use a larger LLM that can perform summarization. Prompt-SAW (Ali et al., 2024) uses a 7B Vicuna model (Chiang et al., 2023) to create a knowledge graph with the key entities and their relationships. Then, each entity-relation pair is encoded with an embedding model and similarity search is performed with the question embedding to determine the most relevant information to keep.

2.3. Extractive Compression

Extractive compression selects relevant documents, sentences, or phrases from the original context. RECOMP also has an extractive compression method that is used to extract the most relevant sentences given the initial context and question. RECOMP trains an encoder model so that useful sentences have higher inner product with the question in the embedding space. In their evaluation, the encoder is fine-tuned from a contriever (110M) checkpoint (Izacard et al., 2021). Document rerankers perform a similar function to RECOMP's extractive compressor. Reranker models take a question and document and output a relevance score for the document to the query. Rerankers are typically applied in RAG pipelines after an initial retrieval step to further refine the document set. Prior work (Nogueira & Cho, 2019) fine-tunes a BERT model (Devlin et al., 2019) for passage rereranking. There is also a line of work (Pradeep et al., 2023a;b) that fine-tunes 7B language models to perform zero-shot listwise reranking. An illustration of extractive compression and its comparison to abstrative compression and token pruning can be found in Figure 3.

2.4. Query-Aware vs Query-Agnostic Compression

Prompt compression methods may further be classified as query-aware or query-agnostic. Query-aware compression methods compress contexts differently depending on the question or task, while query-agnostic compression methods do not rely on the question or task and thus compression may be performed offline only once. For an illustrative comparison, see Figure 4. LLMLingua-2 (Pan et al., 2024) performs query-agnostic compression by training a classifier model to identify and remove redundant tokens. Prompt-

SAW (Ali et al., 2024) also has a query-agnostic variant in which similar information elements in the constructed knowledge graph are de-duplicated.

3. Experiments

To systematically evaluate the various prompt compression methods, we set up experiments designed to measure their effectiveness across a range of scenarios, such as single-document QA, multi-document QA, and summarization.

3.1. Setup

Models: We use Mixtral 8x7B Instruct (Jiang et al., 2024) as the primary LLM. All experiments are conducted with temperature zero and greedy decoding. We also include the results of the same set of experiments with GPT-3.5-Turbo (0613 release) and DBRX Instruct (Team, 2024) in Appendix C.

Datasets: We evaluate using the LongBench benchmark (Bai et al., 2023), which includes tasks requiring reasoning over large contexts. We use nine datasets in total: three for single-document QA, three for multi-document QA, and three for summarization. We follow LongBench's evaluation scripts, using F1 for QA tasks and ROUGE (Lin, 2004) for summarization. Additional dataset details can be found in Appendix B.

Chunking: In this study, *chunking* refers to the process of dividing the large input context into smaller, manageable segments, referred to as *chunks*. In our experiments, unless otherwise specified, each chunk consists of approximately 128 tokens and is carefully constructed to ensure that sentence boundaries are preserved. Chunking is crucial for methods like reranking and LongLLMLingua which operate on coarse-grained units of text by allowing each chunk to be treated as an independent document and assessed independently for its relevance to the query. The terms *chunk* and *document* are used interchangeably in our experiments.

3.2. Evaluated Methods

To evaluate extractive compression, abstractive compression, and token pruning, we selected the following methodologies to evaluate.

Original: We send the whole prompt to the LLM and truncate to the context window if necessary. All compression rates for other methods are reported relative to the compression rate of this method.

LongLLMLingua: We use LongLLMLingua with their suggested hyper-parameters. We vary the rate hyper-parameter to achieve different compression rates. We use a 137M GPT-2 (Radford et al., 2019) as the compressor. LongLLMLingua first prunes irrelevant chunks and then performs token pruning on the kept chunks.

Reranker: We use mxbai-rerank-large-v1 (Shakir et al., 2024) as a reranker model. Given a question, we use the reranker to select the most relevant chunks. We vary the number of selected chunks to achieve different compression rates.

Reranker + LongLLMLingua: We replace LongLLMLingua's coarse-grained document pruning stage with a reranker model. Then we perform token pruning with LongLLMLingua's token pruning methodology. We vary the rate hyper-parameter to achieve different compression rates and otherwise use the recommended hyper-parameters. We use GPT-2 as the compressor for LongLLMLingua's token pruning method.

Reranker + Token Pruning: We implement a custom token pruning method by modifying the reranker so that it performs token-pruning while determining the relevance score for the document. As the reranker is a DeBERTa (He et al., 2020) model, we prune a fixed percentage of document tokens at each layer using attention scores. We prune document tokens that have the lowest attention score with respect to the query tokens. Our custom token pruning method compresses the initial chunk by 20% by pruning 2% of tokens in each of the last 10 layers. The number of chunks selected by the reranker is varied to achieve different compression rates.

Query-Agnostic Abstractive Compression: We use Mistral 7B Instruct (Jiang et al., 2023a) as an abstractive LLM to summarize each chunk offline. For a user query, the reranker first selects relevant chunks and then concatenates the summaries of selected chunks to use as input for the LLM. We ask the summarizer model to compress each chunk by 50% and vary the overall compression rate by varying the number of initially selected chunks in the reranking phase. We show the summarization prompt in Appendix E.

3.3. Main Results

The main results for Mixtral 8x7B Instruct are shown in Figure 1. We include results for GPT-3.5-Turbo and DBRX Instruct in Appendix C and note that they observe similar trends to Mixtral 8x7B Instruct.

3.3.1. Extractive Compression

Extractive compression methods, represented by the reranker, show strong performance across all datasets. On Qasper, the reranker compresses $1.46\times$ and increases accu-

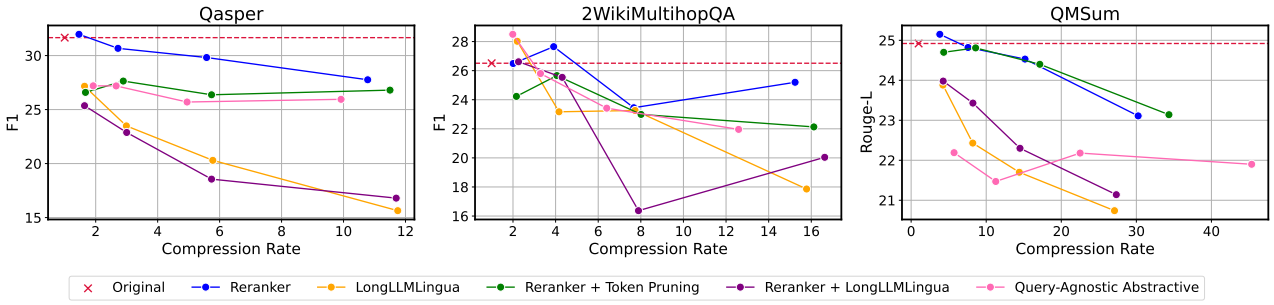

Figure 1. Performance of various compression methods on Qasper, 2WikiMultihopQA, and QMSum datasets with Mixtral 8x7B Instruct. For each dataset, the corresponding graphs plot the accuracy metric—either F1 or Rouge-L—against the compression rate. Results on all nine datasets as well as with GPT-3.5-Turbo and DBRX Instruct can be found in Appendix C.

racy by 0.31 points. On 2WikiMultihopQA it compresses $3.91\times$ while increasing accuracy by 1.14 points. Even at a $15.24\times$ compression rate, it drops just 1.31 points in accuracy. On QMSum, it is able to increase accuracy by 0.23 points while compressing $3.82\times$. A key benefit of extractive compression is the preservation of grammatical constructs due to coarse granularity pruning. This is in stark contrast to unstructured token pruning which can produce incoherent text. The accuracy gains seen at small compression ratios is due to less irrelevant information being provided to the model, mitigating hallucinations and the lost-in-the-middle effect (Liu et al., 2024). Overall, the ability of extractive compression to perform well in a variety of settings makes it a strong candidate for applications.

3.3.2. Abstractive Compression

Abstractive compression methods often lag behind extractive methods in performance. The primary challenge with abstractive compression arises from the use of smaller, potentially weaker models, which may omit crucial information or introduce hallucinations. This is particularly problematic in summarization tasks where the large model has to generate a summary based only on the weaker model's summaries, which can potentially discard information that the large model would have preferred to keep. In our experiments, this problem leads to an accuracy drop of 3.45 points on QMSum at a $11.27\times$ compression rate. Similarly, on QA datasets, we observe performance degradations where query-agnostic abstractive compression performs much worse than extractive compression with the reranker. Therefore online query-aware abstractive compression, as shown in Section 4.2, or fine-tuned summarizers may perform better than prompting out-of-the-box LMs for offline summarization.

3.3.3. Token Pruning

There are three token pruning methods: LongLLMLingua, reranker + LongLLMLingua, reranker + token pruning. We observe that LongLLMLingua and reranker + LongLLMLingua typically exhibit the worst behavior across datasets. In Appendix D, we perform a sweep over LongLLMLingua hyper-parameters but do not see any significant improvement. Reranker + token pruning generally trails slightly behind the plain reranker method. We hypothesize that the lackluster performance of token pruning is due to the disruption of grammar and sentence comprehension caused by unstructured pruning. However, we notice that the reranker + token pruning method achieves competitive results with the reranker on QMSum, achieving even higher accuracy past $30\times$ compression. In contrast, on the question-answering datasets, reranker + token pruning is noticeably worse than just the reranker. In general, token pruning methods appear better suited for aggregation-style tasks that require pieces of knowledge from all segments of the initial context. Furthermore, rather than using out-of-the-box language models, practitioners may see better results by training language models specifically for token pruning (Jung & Kim, 2023; Pan et al., 2024).

4. Additional Analysis

This section details our evaluations on the effects of varying chunk sizes and query-aware abstractive compression. We refer readers to the Appendix for a more comprehensive suite of additional studies on other models and datasets.

4.1. Impact of Chunk Size

To assess the impact of chunk size, we conduct experiments varying chunk size from 128 to 512 tokens. As illustrated in Figure 14, larger chunk sizes perform poorly at high compression rates compared to smaller chunk sizes. For instance, the performance on Qasper with a chunk size of 512 is 4 points lower than chunk sizes of 64, 128, and 256 at approximately $11\times$ compression. When using larger chunk sizes, fewer chunks are generated from the initial context. Consequently, the model might miss important details and

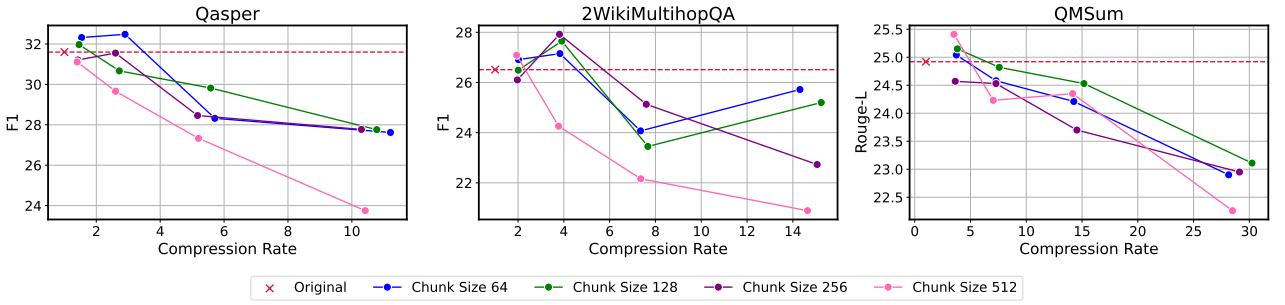

Figure 2. Impact of chunk size on the reranker performance with Mixtral 8x7B Instruct on Qasper, 2WikiMultihopQA, and QMSum datasets. Chunk size is varied between 64, 128, 256, and 512 tokens, and sentence boundaries are respected. Results with GPT-3.5-Turbo can be found in Appendix I.

nuances spread across different parts of the context, leading to a decrease in performance. At smaller compression rates, the chosen chunk size has lesser impact. However, choosing too small chunk sizes risks producing incoherent chunks. Therefore, chunk size should be carefully determined based on the application's data source and desired compression rate.

4.2. Query-Aware Abstractive Compression

The abstractive compression method presented in Section 3.3.2 performs query-agnostic abstractive compression. This is largely beneficial for applications that need low-latency responses, as summaries are precomputed offline. However, it is also possible to perform query-aware abstractive compression, in which summaries are generated by conditioning on the question. Specifically, we use the reranker model to first select relevant chunks and then use Mistral 7B Instruct to summarize the concatenation of selected chunks. We have the reranker select eight chunks, as our experiments indicate that asking the model to summarize more chunks degraded performance. We hypothesize that this diminishing performance is likely due to the difficulty in maintaining coherence and context relevance across larger numbers of chunks. Additionally, after observing difficulties in prompting such models to produce summaries of specific lengths, we used prompting methods similar to RECOMP (Xu et al., 2023), which allows the Mistral model to freely choose the summarization length. In general, our experience with abstractive compression indicates that strong prompt engineering is necessary to achieve desired performance. The summarization prompts are shown in Appendix E.

As shown in Table 1, query-aware abstractive compression demonstrates stronger performance than query-agnostic abstractive compression. We notice extremely strong performance on multi-document question-answering with query-aware compression. We achieve an 8.24 point accuracy increase on 2WikiMultihopQA with a compression rate of $31.32\times$, significantly outperforming both the reranker and query-agnostic abstractive summarization methods. Specifically, query-aware does over 10 points better than query-agnostic compression on 2WikiMultihopQA. On Qasper, query-aware performs 2 points better than query-agnostic compression, and performs 1 point better on QMSum. Therefore, query-aware abstractive compression may be a promising technique for applications willing to handle the overhead of performing on-the-fly summarization.

Method	Qasper		2WikiMultihopQA		QMSum	
	Acc ↑	CR ↑	Acc ↑	CR ↑	Acc ↑	CR ↑
Original	**31.66**	1.00×	26.51	1.00×	**24.92**	1.00×
Mistral 7B Query-Agnostic	23.63	20.07×	24.48	25.02×	20.92	90.77×
Mistral 7B Query-Aware	25.65	**21.25**×	**34.75**	**31.32**×	21.88	**103.89**×

Table 1. Query-aware compared to query-agnostic abstractive compression with Mistral 8x7B Instruct. Mistral 7B is used to produce summaries. Results across four additional datasets and GPT-3.5-Turbo can be found in Appendix J.

5. Conclusions

This study has comprehensively characterized and evaluated a broad spectrum of existing prompt compression methods, which have become critical for long-context inference systems. Our findings reveal that extractive compression is a strong choice, often being able to compress over $10\times$ with minimal accuracy loss. Token pruning demonstrates marginal improvements over extractive compression on summarization tasks. Furthermore, the performance of abstractive compression can be significantly enhanced, by up to 10 points in multi-document QA tasks at $30\times$ compression, through the generation of query-aware summaries. Ultimately, this study not only sheds light on the varied efficacy of prompt compression strategies but also sets the stage for innovative advancements in optimizing applications relying on long context inference.

References

Ali, M. A., Li, Z., Yang, S., Cheng, K., Cao, Y., Huang, T., Hu, L., Yu, L., and Wang, D. Prompt-saw: Leveraging relation-aware graphs for textual prompt compression. *arXiv preprint arXiv:2404.00489*, 2024.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. A dataset of information-seeking questions and answers anchored in research papers, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Fabbri, A. R., Li, I., She, T., Li, S., and Radev, D. R. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model, 2019.

Goyal, S., Choudhury, A. R., Raje, S., Chakaravarthy, V., Sabharwal, Y., and Verma, A. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pp. 3690–3699. PMLR, 2020.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Ho, X., Nguyen, A.-K. D., Sugawara, S., and Aizawa, A. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. Efficient attentions for long document summarization, 2021.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023a.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023b.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*, 2023c.

Jung, H. and Kim, K.-J. Discrete prompt compression with reinforcement learning. *arXiv preprint arXiv:2308.08758*, 2023.

Kim, G. and Cho, K. Length-adaptive transformer: Train once with length drop, use anytime with search. *arXiv preprint arXiv:2010.07003*, 2020.

Kim, S., Shen, S., Thorsley, D., Gholami, A., Kwon, W., Hassoun, J., and Keutzer, K. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 784–794, 2022.

Kim, S., Hooper, C., Wattanawong, T., Kang, M., Yan, R., Genc, H., Dinh, G., Huang, Q., Keutzer, K., Mahoney, M. W., Shao, Y. S., and Gholami, A. Full stack optimization of transformer inference: a survey, 2023.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The narrativeqa reading comprehension challenge, 2017.

Li, Y., Dong, B., Lin, C., and Guerin, F. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*, 2023.

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Nogueira, R. and Cho, K. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Packer, C., Fang, V., Patil, S. G., Lin, K., Wooders, S., and Gonzalez, J. E. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Pan, Z., Wu, Q., Jiang, H., Xia, M., Luo, X., Zhang, J., Lin, Q., Rühle, V., Yang, Y., Lin, C.-Y., Zhao, H. V., Qiu, L., and Zhang, D. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression, 2024.

Pradeep, R., Sharifymoghaddam, S., and Lin, J. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*, 2023a.

Pradeep, R., Sharifymoghaddam, S., and Lin, J. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*, 2023b.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

Shakir, A., Koenig, D., Lipp, J., and Lee, S. Boost your search with the crispy mixedbread rerank models, 2024. URL https://www.mixedbread.ai/blog/mxbai-rerank-v1.

Tan, S., Li, X., Patil, S., Wu, Z., Zhang, T., Keutzer, K., Gonzalez, J. E., and Popa, R. A. Lloco: Learning long contexts offline. *arXiv preprint arXiv:2404.07979*, 2024.

Team, T. M. R. Introducing dbrx: A new state-of-the-art open llm, 2024. URL https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Musique: Multihop questions via single-hop question composition, 2022.

Wang, H., Zhang, Z., and Han, S. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 97–110. IEEE, 2021.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., and Mann, G. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

Xiao, W. and Carenini, G. Extractive summarization of long documents by combining global and local context, 2019.

Xu, F., Shi, W., and Choi, E. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*, 2023.

Yang, H., Li, Z., Zhang, Y., Wang, J., Cheng, N., Li, M., and Xiao, J. Prca: Fitting black-box large language models for retrieval question answering via pluggable reward-driven contextual adapter. *arXiv preprint arXiv:2310.18347*, 2023a.

Yang, H., Liu, X.-Y., and Wang, C. D. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*, 2023b.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.

Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A. H., Celikyilmaz, A., Liu, Y., Qiu, X., and Radev, D. Qmsum: A new benchmark for query-based multi-domain meeting summarization, 2021.

A. Prompt Compression Methods

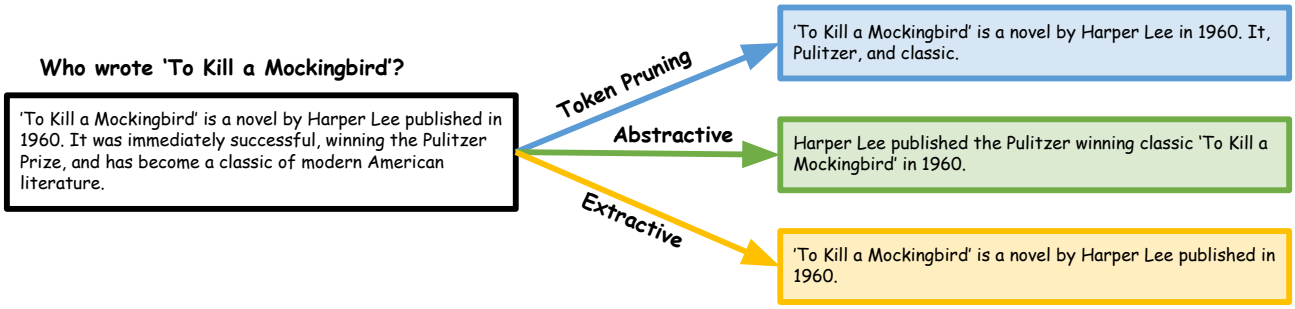

Figure 3. An illustration of different prompt compression methods. Token pruning methods like LongLLMLingua (Pan et al., 2024), Selective-Context (Li et al., 2023), and PCRL (Jung & Kim, 2023) perform compression by discarding irrelevant tokens; extractive compression methods like RECOMP (Xu et al., 2023) and reranker-based compression select documents, sentences, or phrases from the original context without altering them; abstractive compression methods like Prompt-SAW (Ali et al., 2024), RECOMP, and PRCA (Yang et al., 2023a) generate summaries by synthesizing information. In this example, each of the methods compresses the original context while keeping the necessary information to determine the book's author.

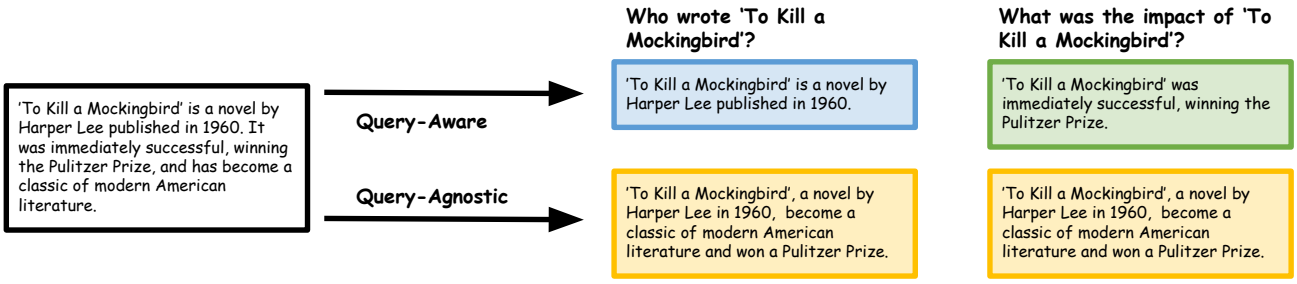

Figure 4. An illustration of query-aware and query-agnostic compression applied to a document in the prompt context. With query-aware compression, the compressed context of the document changes based on the user's specific query, presenting a tailored version each time. Conversely, query-agnostic compression maintains a consistent compressed context of the document, irrespective of the query presented.

B. LongBench Dataset Details

We give a brief description of each evaluated dataset in LongBench, as well as the average token count measured by GPT-3.5-Turbo's tokenizer.

NarrativeQA (Kočiský et al., 2017): Question-answering over stories. Average tokens: 29,780.

Qasper (Dasigi et al., 2021): Question-answering over NLP papers. Average tokens: 4,923.

MultiFieldQA (Bai et al., 2023): Question-answering over a variety of document types such as legal documents, government reports, and academic papers. Average tokens: 6,938.

HotpotQA (Yang et al., 2018): 2-hop question-answering. Average tokens: 12,793.

2WikiMultihopQA (Ho et al., 2020): Up to 5-hop question-answering. Average tokens: 7,116.

MuSiQue (Trivedi et al., 2022): Up to 4-hop question-answering. Average tokens: 15,577.

GovReport (Huang et al., 2021): Summarization of detailed government reports. Average tokens: 10,242.

QMSum (Zhong et al., 2021): Query-based summarization over meeting notes. Average tokens: 13,855.

MultiNews (Fabbri et al., 2019): Summarization of multiple news articles. Average tokens: 2,609.

C. Full Mixtral 8x7B, GPT-3.5-Turbo, and DBRX Instruct Results

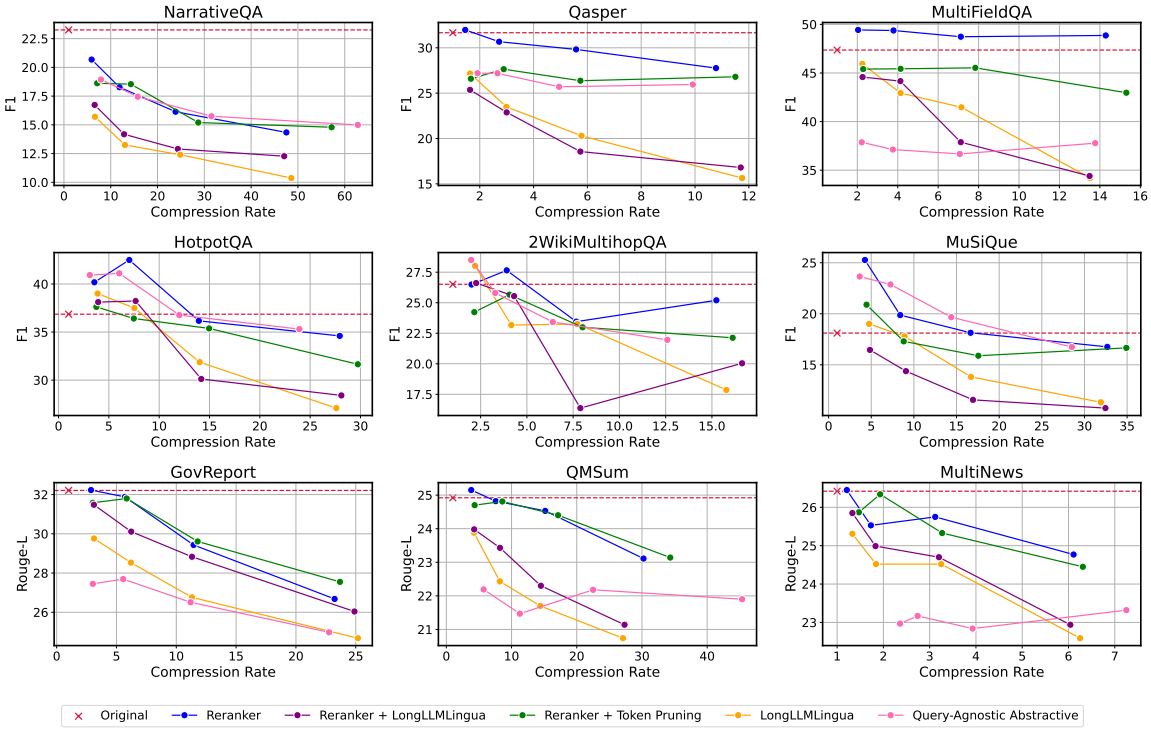

Figure 5. Performance of various compression methods on all nine datasets from LongBench with Mixtral 8x7B Instruct model. For each dataset, the corresponding graphs plot the accuracy metric—either F1 or Rouge-L—against the compression rate.

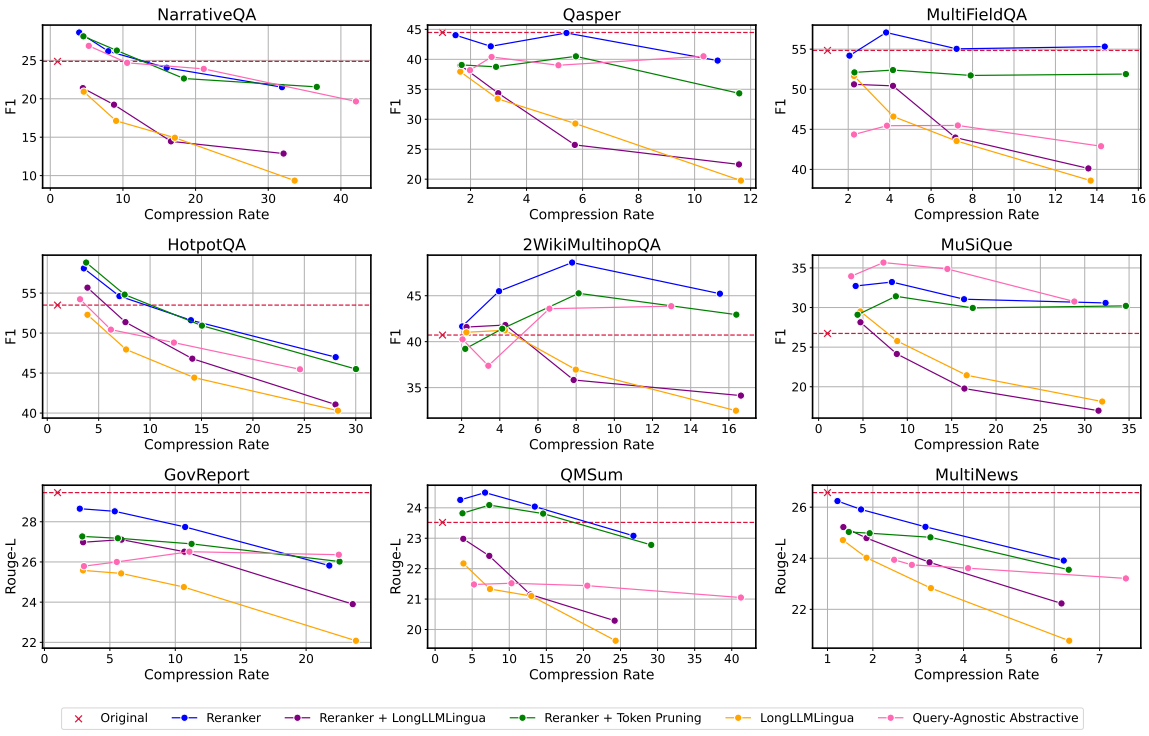

Figure 6. Performance of various compression methods on nine datasets from LongBench with GPT-3.5-Turbo. For each dataset, the corresponding graphs plot the accuracy metric—either F1 or Rouge-L—against the compression rate.

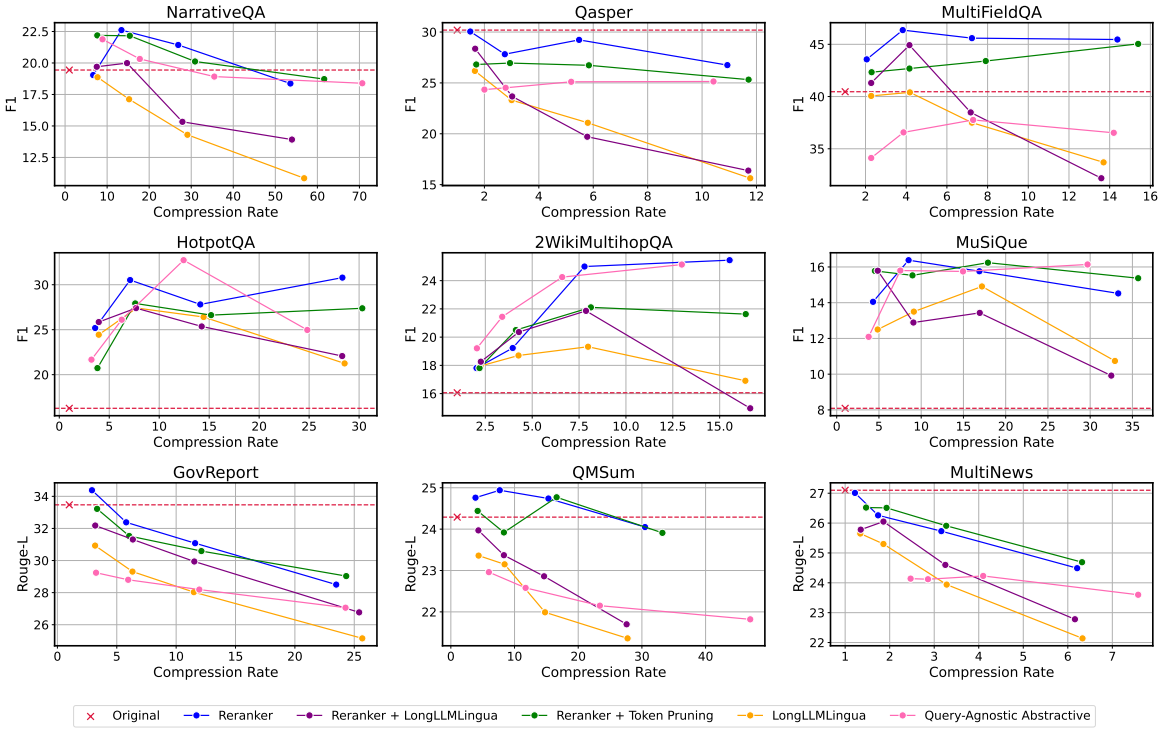

Figure 7. Performance of various compression methods on nine datasets from LongBench with DBRX Instruct. For each dataset, the corresponding graphs plot the accuracy metric—either F1 or Rouge-L—against the compression rate.

D. LongLLMLingua Hyper-Parameter Sweep

In Section 3.3.3, we used hyper-parameters for LongLLMLingua as recommended by the authors. Here, we perform a study where we sweep over 8 different hyper-parameter configurations for LongLLMLingua. We conduct the study on both Mixtral 8x7B Instruct and GPT-3.5-Turbo, showing the results on NarrativeQA, HotpotQA, and MultiNews. Unlike Mixtral 8x7B Instruct and DBRX Instruct, GPT-3.5-Turbo is not deterministic at these settings. Therefore, for all experiments with GPT-3.5-Turbo, we report averages over three trials. GPT-3.5-Turbo has a context window of 16k tokens, and both Mixtral 8x7B Instruct and DBRX Instruct have context windows of 32k tokens.

For the main results, we use the following hyper-parameters with LongLLMLingua. Sentence-level filtering turned off, dynamic context compression rate is set to 0.3 context budget is set to $+100$, condition in question is set to "after_condition", reorder context is set to "sort", and condition compare is set to true. All other hyper-parameters are otherwise default. For the LongLLMLingua hyper-parameter sweep, we toggle the use of sentence-level filtering and we vary the dynamic context compression rate between 0, 0.2, 0.3, and 0.4. As shown in Figure 8 and Figure 9, our chosen hyper-parameters perform well and all tested configurations exhibit similar trends.

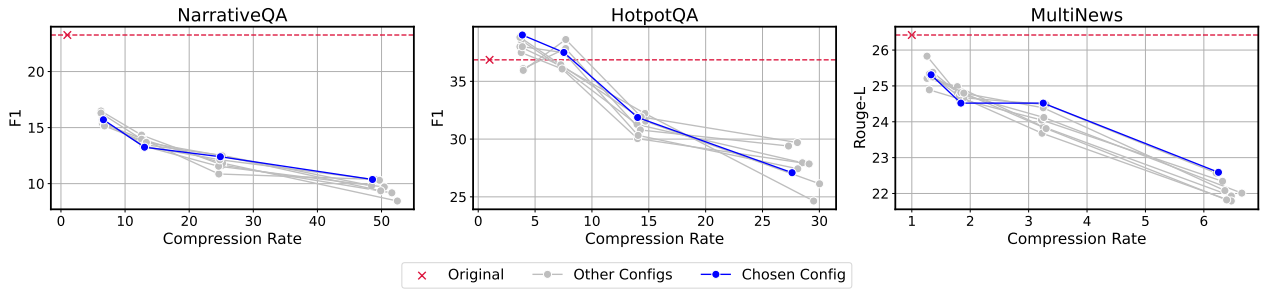

Figure 8. Evaluation of the Mixtral 8x7B Instruct model with varying LongLLMLingua hyper-parameters on NarrativeQA, HotpotQA, and MultiNews datasets. Performance is analyzed with adjustments to the dynamic context compression rate and sentence-level filtering.

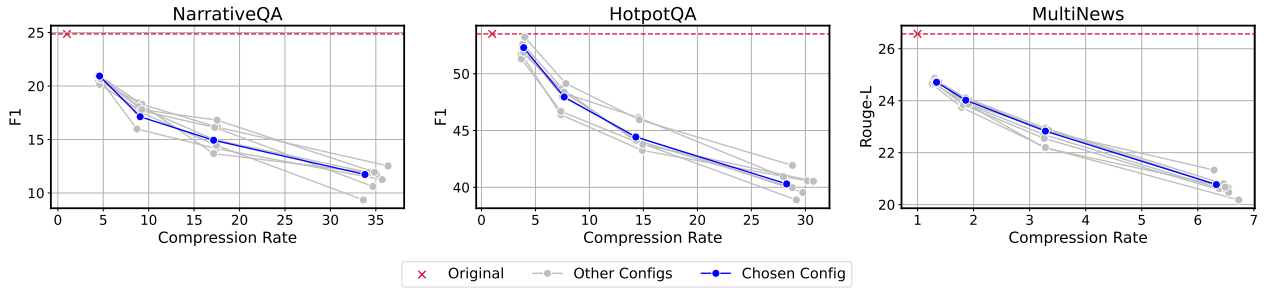

Figure 9. Evaluation of the GPT-3.5-Turbo model with varying LongLLMLingua hyper-parameters on NarrativeQA, HotpotQA, and MultiNews datasets. Performance is analyzed with adjustments to the dynamic context compression rate and sentence-level filtering.

E. Abstractive Compression Prompts

In Table E, we show the prompts used to perform query-aware and query-agnostic abstractive compression.

Method	Prompt
Query-Agnostic	Could you please rephrase the paragraph to make it short, and keep 50% tokens. Respond with ONLY the compressed paragraph and nothing else. Paragraph: `paragraph`
Query-Aware (Mistral 7B Instruct)	Compress the information in the retrieved documents into a summary that could be used to answer the question: Question: `query` Retrieved documents: `docs`
Query-Aware (LLaMA 3 8B Instruct)	Compress the information in the retrieved documents into a summary that could be used to answer the question. Do NOT try to directly answer the question. Question: `query` Retrieved documents: `docs`

Table 2. Prompts used for query-aware and query-agnostic abstractive compression.

F. Effect of Weaker Reranker

In Section 3.3, we used mxbai-rerank-large-v1 (435M) as the reranker. We perform a study when using a weaker reranker model, namely mxbai-rerank-base-v1 (184M). Since mxbai-rerank-base-v1 only has 12 layers, we modify our custom token pruning scheme to prune by 4% starting from layer 8. We conduct the study on GPT-3.5-Turbo and Mixtral 8x7B using MultiFieldQA, MuSiQue, and MultiNews. As shown in Figure 10 and Figure 11, the large reranker generally outperforms the base reranker across all three datasets. However, there are certain points at which the base reranker outperforms the large reranker. Thus the base reranker proves to be a suitable alternative in resource constrained settings.

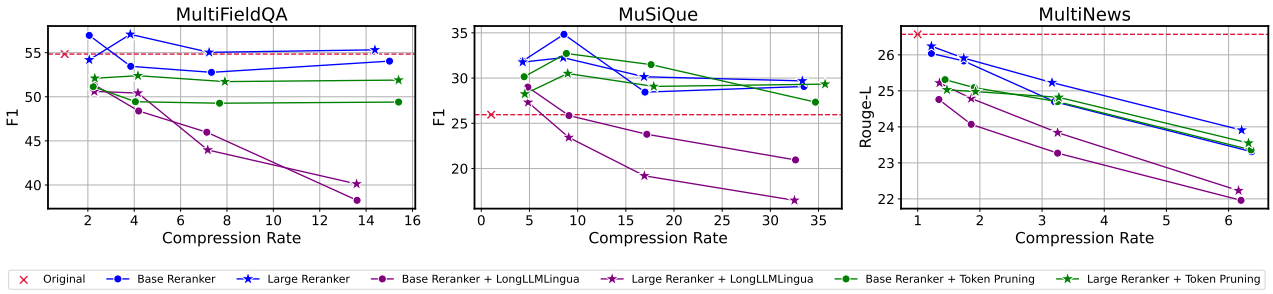

Figure 10. Performance comparison between using mxbai-rerank-large-v1 (435M) versus mxbai-rerank-base-v1 (184M) with GPT-3.5-Turbo as the LLM. Results are shown on MultiFieldQA, MuSiQue, and MultiNews.

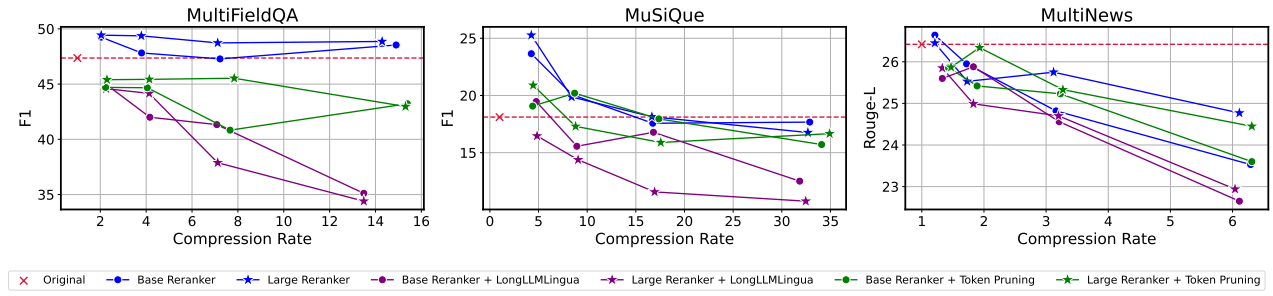

Figure 11. Performance comparison between using mxbai-rerank-large-v1 (435M) versus mxbai-rerank-base-v1 (184M) with Mixtral 8x7B as the LLM. Results are shown on MultiFieldQA, MuSiQue, and MultiNews.

G. Retriever vs Reranker

Instead of using a reranker for chunk-level compression, it is also possible to prune irrelevant chunks by using similarity search between the question and chunk embeddings. We conduct the study on GPT-3.5-Turbo, using OpenAI's text-embedding-3-small as the embedder, and show the results on MultiFieldQA, 2WikiMultihopQA, and QMSum. As

shown Figure 12, the reranker outperforms the retriever model. However, the increased accuracy comes at the cost of increased compute requirements at run time from running the reranker model over the chunks, compared to the cheap similarity search performed after embedding the question with the embedding model. In many settings, reranking is applied after an initial retrieval step to reduce the number of documents that need to be reranked.

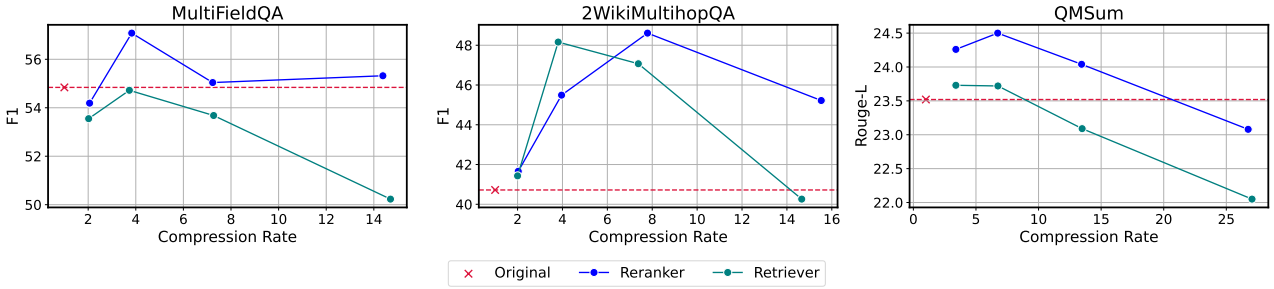

Figure 12. Analysis of performing extractive compression using standard retrieval over embedding space compared to reranking. For retrieval, embeddings are produced using text-embedding-3-small. GPT-3.5-Turbo is used as the LLM and evaluated on MultiFieldQA, 2WikiMultihopQA, and QMSum.

H. Aggressive Token Pruning

For the token pruning methods in Section 3.3, the reranker selects 25% more chunks than originally and then applied a token pruning rate of 20% to achieve each compression ratio. Here, we perform a study where the reranker selects $2\times$ more chunks and an aggressive token pruning rate of 50% is applied. We conduct the study on GPT-3.5-Turbo and show the results on NarrativeQA, MuSiQue, and GovReport. As shown in Figure 13, such aggressive token pruning leads to accuracy degradation. After observing the pruned context, we hypothesize that this is because aggressive token pruning leads to unstructured text that does not respect grammatical constructs, making it difficult for the downstream model to correctly reason over it.

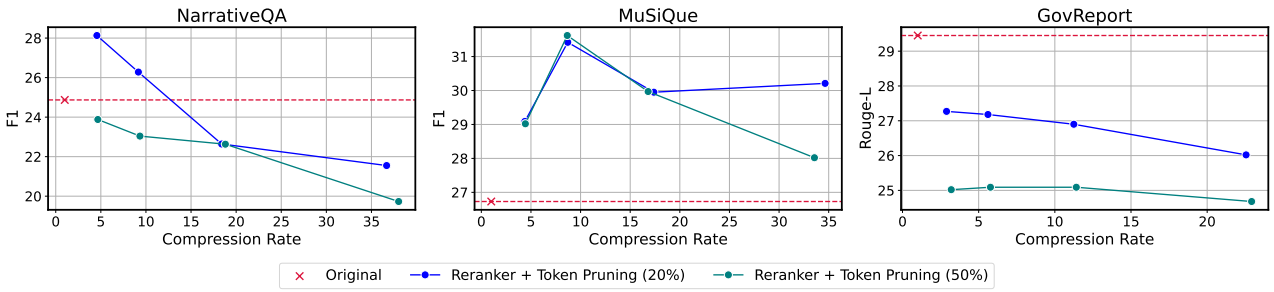

Figure 13. Performance analysis of using aggressive token pruning. We compare the original token pruning method which prunes 20% of the tokens to a token pruning method that prunes 50% of the tokens. GPT-3.5-Turbo is used as the LLM and evaluated on NarrativeQA, MuSiQue, and GovReport.

I. Impact of Chunk Size on GPT-3.5-Turbo

To determine the impact of chunk size, we run a set of experiments after changing chunk size from 128 to 512 tokens. We show the results on Qasper, 2WikiMultihopQA, and QMSum using GPT-3.5-Turbo as the LLM. The results are shown in Figure 14 and Figure 15. We notice that large chunk sizes do not perform well at large compression ratios when compared to smaller chunk sizes. We hypothesize that this is because there are very few chunks being provided to the model when the chunk size is large. As a result, the model does not have the ability to see text from varying regions of the initial context. In

contrast, using smaller chunk sizes allows more chunks to be used, alleviating this issue. At smaller compression ratios, the chosen chunk size has lesser impact.

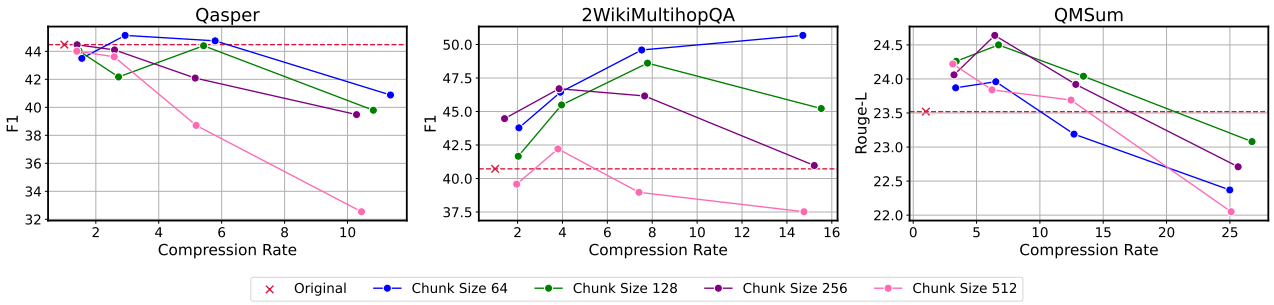

Figure 14. Impact of chunk size on the reranker with GPT-3.5-Turbo. Chunk size is varied between 64, 128, 256, and 512 tokens. Sentence boundaries are respected. Results are shown on Qasper, 2WikiMultihopQA, and QMSum.

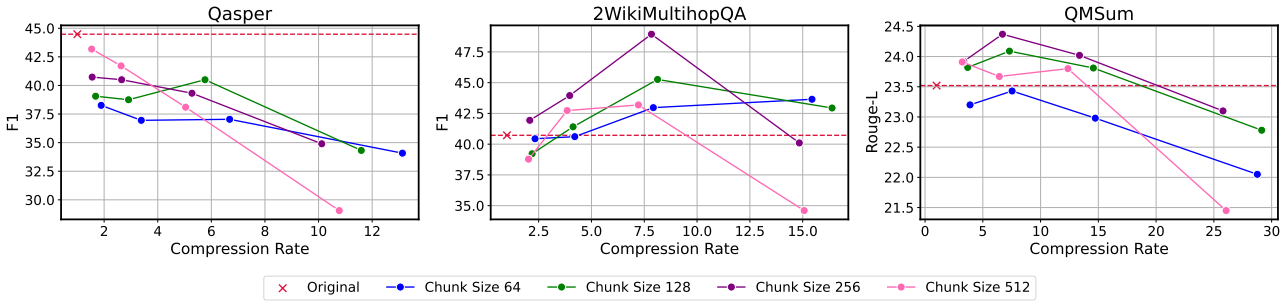

Figure 15. Impact of chunk size on the token pruning reranker with GPT-3.5-Turbo. Chunk size is varied between 64, 128, 256, and 512 tokens. Sentence boundaries are respected. Results are shown on Qasper, 2WikiMultihopQA, and QMSum.

J. Full Query-Aware Abstractive Compression Results

In Table 3 and Table 4, we show the results of query-aware compression on seven of the LongBench datasets, with both GPT-3.5-Turbo and Mixtral 8x7B. We also show the results with Mistral 7B and LLaMA 3 8B as summarizers. Our experiments indicate that it is difficult to control the length of summaries, making the compression rate for query-aware abstractive compression difficult to predict.

Table 3. Query-aware abstractive compression results with GPT-3.5-Turbo. We use Mistral 7B Instruct and LLaMA-3 8B Instruct to generate summaries from chunks selected by the reranker.

Method	NQA		QAS		MFE		HQA		WMQA		MSQ		QMS	
	Acc	CR	Acc	CR	Acc	CR	Acc	CR	Acc	CR	Acc	CR	Acc	CR
Original	24.87	1.00×	44.48	1.00×	54.84	1.00×	53.5	1.00×	40.72	1.00×	26.73	1.00×	23.52	1.00×
Mistral 7B														
8 chunks	20.48	104.09×	38.36	21.62×	46.20	31.24×	49.15	46.19×	51.37	30.55×	30.71	65.14×	20.99	87.10×
16 chunks	25.56	86.12×	36.27	19.96×	47.80	28.14×	52.23	44.36×	47.63	25.31×	33.75	58.28×	21.21	76.07×
32 chunks	24.12	74.44×	31.70	19.68×	46.47	27.50×	50.47	44.47×	47.93	22.79×	30.49	52.57×	20.96	62.75×
LLaMA 3 8B														
8 chunks	20.14	124.03×	40.86	35.90×	47.25	54.34×	48.10	112.60×	47.10	61.56×	26.56	124.11×	22.06	94.28×
16 chunks	23.07	106.00×	38.36	28.71×	47.35	44.59×	48.81	91.69×	45.38	49.48×	28.83	103.24×	21.33	77.22×
32 chunks	21.97	75.54×	33.88	23.89×	40.30	33.16×	47.18	64.31×	42.64	31.70×	30.45	70.56×	20.68	59.87×

Table 4. Query-aware abstractive compression results with Mixtral 8x7B Instruct. We use Mistral 7B Instruct and LLaMA-3 8B Instruct to generate summaries from chunks selected by the reranker.

Method	NQA		QAS		MFE		HQA		WMQA		MSQ		QMS	
	Acc	CR	Acc	CR	Acc	CR	Acc	CR	Acc	CR	Acc	CR	Acc	CR
Original	23.26	1.00×	31.66	1.00×	47.36	1.00×	36.86	1.00×	26.51	1.00×	18.11	1.00×	24.92	1.00×
Mistral 7B														
8 chunks	15.65	165.39×	25.65	21.25×	42.29	32.26×	38.01	53.95×	34.75	31.32×	19.81	66.88×	21.88	103.89×
16 chunks	15.34	135.35×	23.62	19.68×	44.82	28.25×	38.11	47.34×	29.79	27.27×	21.53	59.21×	21.16	90.03×
32 chunks	18.17	118.15×	19.86	20.21×	40.86	28.00×	39.84	44.85×	30.82	26.75×	19.68	55.93×	21.08	78.37×
LLaMA 3 8B														
8 chunks	14.13	197.32×	25.82	35.58×	41.72	53.79×	34.83	116.14×	28.61	63.79×	16.17	135.038×	22.06	112.08×
16 chunks	16.21	167.87×	24.53	28.64×	42.90	45.19×	39.20	96.06×	27.93	50.03×	20.99	106.54×	21.14	91.13×
32 chunks	17.66	118.51×	21.87	24.10×	37.80	33.52×	33.35	67.31×	25.05	33.62×	15.84	75.27×	21.40	70.54×