HiRAG: Human-inspired Retrieval-Augmented Generation

Enzo Ruedas, Baptiste Pouthier

NXP Semiconductors, Sophia-Antipolis, France
enzo.ruedas@nxp.com, baptiste.pouthier@nxp.com

Abstract

Retrieval-Augmented Generation effectively overcomes Large Language Models knowledge limitations. This approach improves answer quality by incorporating additional context into the input prompt. In this paper, we propose the Human-inspired Retrieval-Augmented Generation (HiRAG) that reduces prompt sizes by creating and retrieving short yet comprehensive contexts. Our experiments demonstrate that HiRAG improves retrieval accuracy with smaller embedding models, especially for technical data. On average, HiRAG condenses more knowledge into shorter prompts while preserving context quality. This reduction lowers latency during inference, making it well-suited for embedded devices.

Introduction

Large Language Models (LLMs) facilitate deep and natural interactions between humans and machines. Pre-trained foundation models are often customized to improve performance in specific use cases. Unlike fine-tuning approaches (Hu et al. 2022) that modify the model's parameters, Retrieval-Augmented Generation (RAG) (Lewis et al. 2020) enhances response quality by enriching the LLM's prompt (Boros et al. 2024). RAG with factual context also reduces false answers, called hallucinations. RAG consists of three phases: ingestion, retrieval, and generation. This study focuses on the first two phases.

The ingestion phase involves segmenting a large textual knowledge into short pieces of text named chunks. Each chunk is stored with a vector embedding representing its meaning. Fixed-Size (FS) chunking (Venish and Siva Sankar 2016) was introduced to divide text into overlapping segments of predefined length. As this method overlooks human language structure, content-aware strategies such as Recursive chunking (Rec.) (Gong et al. 2020) were developed. In practice, the text is recursively cut on punctuation signs, until a desired number of characters is reached. Natural Language Processing libraries such as NLTK (Bird 2006) and SpaCy (Neumann et al. 2019) offer advanced tools for generating chunks with better context preservation.

Presented at the Workshop on Preparing Good Data for Generative AI: Challenges and Approaches (Good-Data) in conjunction with AAAI 2025. The authors retain the copyright.

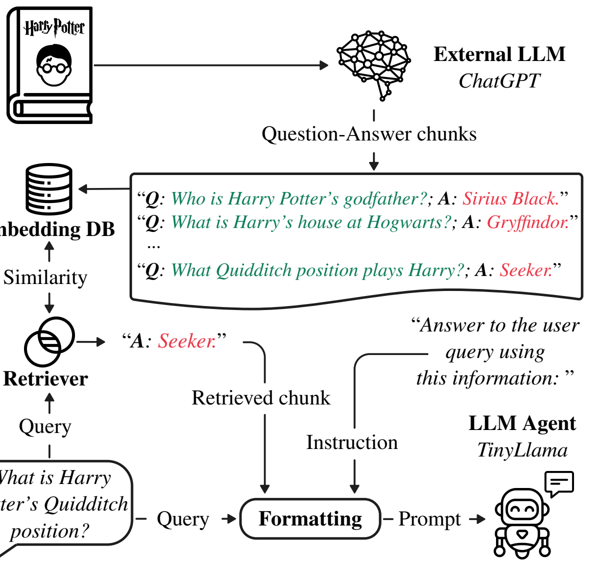
Harry Potter
External LLM
ChatGPT
Question-Answer chunks
"Q: Who is Harry Potter's godfather?; A: Sirius Black."
"Q: What is Harry's house at Hogwarts?; A: Gryffindor."
...
"Q: What Quidditch position plays Harry?; A: Seeker."
Embedding DB
Similarity
Retriever
"A: Seeker."
Retrieved chunk
"Answer to the user query using this information: "
Instruction
LLM Agent
TinyLlama
Query
What is Harry Potter's Quidditch position?
Query
Formatting
Prompt

Figure 1: In HiRAG, an external LLM converts large texts into Question-Answer pairs. The user query is compared with all pairs to feed the relevant information to the prompt.

The retrieval phase consists in identifying the most relevant chunks to a query by comparing the similarity of all stored vectors with the query's embedding (Farouk 2020). The most similar chunks are considered the most pertinent pieces of information and are added to the LLM's prompt. Their quality depends on both the chunking strategy and the embedding model's ability to capture textual meaning (Cer et al. 2018). The Massive Text Embedding Benchmark (MTEB) (Muennighoff et al. 2022) evaluates and compares the size and performance of various embedding models across different tasks. For example, the 7-billion parameter SFR-Embedding-Mistral model (Jiang et al. 2023; Wang et al. 2024) excels in tasks such as pair classification, retrieval, and re-ranking. In contrast, the 23-million parameter GIST-all-MiniLM-L6-v2 (Wang et al. 2020; Solatorio 2024) is a much more compact alternative that still achieves impressive results on retrieval tasks (Muennighoff et al. 2022).

However, integrating LLMs into embedded devices remains challenging due to limited memory, restricted data movement, and reduced computational power (Dhar et al. 2024). Therefore, research has focused on smaller LLMs (Abdin et al. 2024) designed for embedded applications. The 1-billion parameter TinyLlama (Zhang et al. 2024), the 7-billion Llama2 (Touvron et al. 2023), or more recently the 8-billion parameter Llama3 (Dubey et al. 2024) offer compact and promising alternatives. Further reducing LLMs' size is possible through structural (Ma, Fang, and Wang 2023) and/or post-training adaptations (Xiao et al. 2023), but entails an additional sacrifice of proficiency for latency. Moreover, RAG complicates the integration process by requiring an extra embedding model. Re-ranking algorithms (Carraro and Bridge 2024) that improve chunk relevancy can also exacerbate latency. While efforts have been made to reduce embedding model sizes, research on adapting RAG for embedded applications remains limited.

In this paper, we present HiRAG which delivers concise yet comprehensive contextual information to the prompt. The phases of the HiRAG process are illustrated in Figure 1. Our contribution is threefold: (1) a relatively more compact textual database, (2) the retrieval of concise and high-quality context information, and (3) smaller prompts to reduce inference latency.

Human-inspired RAG

This paper introduces a RAG pipeline designed for embedded applications, which evolves chunking from basic text segmentation to knowledge-driven summarization. Our approach generates fewer, smaller, but smarter chunks. In the ingestion phase, an advanced LLM such as ChatGPT-3 (Ye et al. 2023) transforms textual data into Question-Answer (QA) pairs that capture all essential information. Question-answering agents typically handle spoken or written questions, whereas RAG databases often contain more technical vocabulary that differs from standard conversational phrasing. Figure 2 shows how leveraging LLMs' understanding and broad knowledge reduces parsing artifacts and produces concise yet exhaustive chunks with high clarity.

We evaluate HiRAG on two configurations for the ingestion phase and two complementary configurations for the retrieval phase.

The two ingestion phase configurations are: (1) Emb_{Q}, where only the question part of the QA chunk is vectorized, and (2) Emb_{QA}, which converts into embedding the entire QA pair. While (1) aims to maximize the similarity between stored embeddings and the user's question, (2) considers that the question's phrasing might appear in both the question and answer part of a chunk.

The two retrieval phase configurations are: (1) Ret_{A}, which adds only the answer part of the retrieved chunk to the prompt, and (2) Ret_{QA}, which inserts the entire QA pair into the prompt. While (1) seeks to minimize prompt length to reduce latency, potentially limiting context understanding, (2) provides clearer and longer context by adding the whole QA pair into the prompt.

Prompt:
Give me as many question-answer pairs as needed to cover the whole following text. Question (Q) and answer (A) must be short and precise. The pairs must be formatted as: Q: ; A: .
With Remote Start Shown, Without Similar: If equipped, the RKE transmitter functions work up to 60 m (197 ft) away from the vehicle. Other conditions can affect the performance of the transmitter. See Remote Keyless Entry (RKE) System 08.
Answer:
- **Q**: How far do the RKE transmitter functions work from the vehicle?; **A**: Up to 60 meters (197 feet) away.
- **Q**: What can affect the performance of the RKE transmitter functions?; **A**: Other conditions.
- **Q**: Where can more information about the Remote Keyless Entry (RKE) System be found?; **A**: In the vehicle's manual.

Figure 2: Example of a prompt for generating QA pairs from a given context paragraph, using ChatGPT-3.

Experimental Setup

Two QA databases were generated using ChatGPT-3 and the 7-billion-parameter Llama2 LLMs, respectively. In practice, textual data is segmented into paragraphs, that are processed by the external LLM, as shown in Figure 2. QA pairs are then extracted directly from the LLM's responses.

Comparing Chunking Strategies

The two HiRAG ingestion configurations Emb_{Q} and Emb_{QA} are compared with four state-of-the-art chunking strategies: FS, Rec., NLTK, and SpaCy. These baselines are evaluated with different chunk lengths and overlaps, both defined by the size in number of characters. Three standard size/overlap pairs were selected: the Small 64/32 (S), the Medium 128/64 (M), and the Large 256/128 (L). In total, 16 chunking strategies are compared in this study.

Comparing Retrieval Strategies

The impact of HiRAG QA reformatting was compared using two embedding models of different sizes: the large and advanced SFR-Embedding-Mistral model, and the compact GIST-all-MiniLM-L6-v2 model. In this study, only the single chunk with the highest cosine similarity was considered to simplify the retrieved context evaluation.

Datasets

Our experiments are conducted on two datasets: the validation set of the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016) and the Car Owner's Manual (COM). These datasets are designed to evaluate RAG across various themes. Each paragraph of the datasets is associated with one or more questions along with their respective answers.

The SQuAD is composed of general knowledge data from Wikipedia. Its validation set has been filtered out from unanswerable questions. After preprocessing, 2,067 paragraphs and 10,569 question-answers remain.

We crafted the COM dataset from a 284-page car user guide which is composed of 356 paragraphs of technical automotive data. 2,104 question-answers were generated

Table 1: Chunking strategies comparison on SQuAD

Ingestion strategy	GIST Acc. ↑	SFR Acc. ↑	#chunk (10^3) ↓	#char. (10^6) ↓
FS (S)	0.44	0.77	251	15.88
FS (M)	0.50	**0.80**	123	15.41
FS (L)	0.51	0.78	59	14.43
Rec. (S)	0.48	0.77	252	14.96
Rec. (M)	0.52	**0.80**	123	14.97
Rec. (L)	0.53	0.78	59	14.21
NLTK (S)	**0.63**	0.76	53	8.19
NLTK (M)	**0.63**	**0.80**	53	8.2
NLTK (L)	0.62	0.77	**43**	8.71
SpaCy (S)	0.62	0.75	53	8.19
SpaCy (M)	**0.63**	0.76	52	8.2
SpaCy (L)	0.62	0.77	**43**	8.70
HiRAG				
Llama2 QA chunks				
Emb_{Q}	0.50	0.67	70	10.49
Emb_{QA}	0.60	0.77		
ChatGPT QA chunks				
Emb_{Q}	0.54	0.70	60	**6.64**
Emb_{QA}	0.62	0.79		

Table 2: Chunking strategies comparison on COM dataset

Ingestion strategy	GIST Acc. ↑	SFR Acc. ↑	#chunk (10^3) ↓	#char. (10^6) ↓
FS (S)	0.41	0.70	15	0.97
FS (M)	0.42	0.74	7	0.96
FS (L)	0.48	**0.75**	3	0.93
Rec. (S)	0.44	0.74	15	0.92
Rec. (M)	0.47	**0.75**	7	0.93
Rec. (L)	0.46	0.74	3	0.91
NLTK (S)	0.52	0.71	5	0.49
NLTK (M)	0.55	0.73	4	0.53
NLTK (L)	0.51	0.73	3	0.69
SpaCy (S)	0.55	0.70	5	0.50
SpaCy (M)	0.56	0.74	4	0.53
SpaCy (L)	0.51	0.74	3	0.70
HiRAG				
Llama2 QA chunks				
Emb_{Q}	0.61	0.71	2	0.45
Emb_{QA}	0.58	0.69		
ChatGPT QA chunks				
Emb_{Q}	**0.64**	0.73	**1**	**0.22**
Emb_{QA}	**0.64**	0.71		

for this dataset using Gemini 1.0 Pro (Reid et al. 2024), following a similar process shown in Figure 2.

For each question in both datasets, the SFR-Embedding-Mistral model was used to identify the most similar chunk from among the available options within the same paragraph. The selected chunks are labeled as the ground truths.

Metrics

The quality of each ingestion strategy is measured by its resulting accuracy (Acc.) in retrieving the ground truth chunk. The database memory impact is assessed by both the chunk count and the character count. The chunk count indicates the number of vectors sorted in the database, while the character count indicates the knowledge compactness after chunking.

The BERTScore (Zhang et al. 2020) metric is calculated by comparing the retrieved chunk with the LLM's expected answer to evaluate relevancy. A higher BERTScore corresponds to a better similarity. The ROUGE (Lin 2004) scores provide complementary insight on the overlap between the words in the expected answer and those in the retrieved chunk. In particular, the ROUGE 1 and 2 scores evaluate the co-occurrence of unigrams and bigrams between two texts, while the L score measures the longest common subsequence.

Experimental Results

The impact of HiRAG is studied for the ingestion and retrieval phases. Different combinations of Emb_{QA}, Emb_{Q}, Ret_{QA}, and Ret_{A} are denoted by the $\wedge$ symbol. For instance, the combination of $\text{Emb}_{\text{QA}} \wedge \text{Ret}_{\text{A}}$ configurations refers to the use of Emb_{QA} for ingestion and Ret_{A} for retrieval.

Ingestion Strategies Evaluation

Table 1 and 2 highlight the chunk database quality for SQuAD and COM datasets, respectively. For each dataset, 16 chunk databases have been generated using the different ingestion strategies. The tables show the accuracy of the two embedding models in finding the ground truth chunk in the database. They also present the total chunk count and the character count, illustrating the storage requirements for the same initial knowledge data.

Table 1 shows consistent accuracy of the SFR-Embedding-Mistral model across different ingestion strategies. The GIST-all-MiniLM-L6-v2 model exhibits more varied accuracy, suggesting that chunk database quality has a greater impact on retrieval accuracy as the embedding model size decreases. On SQuAD, the NLTK (L) and SpaCy (L) strategies generate fewer chunks. In comparison, the HiRAG chunk counts remain fairly low. HiRAG with ChatGPT best condenses the textual data.

Table 2 confirms that the chunk database quality has a greater impact when using a smaller embedding model. On COM dataset, HiRAG demonstrates the best accuracies with scores over 0.61 using GIST-all-MiniLM-L6-v2. Notably, this method also produces fewer and more condensed chunks. The HiRAG using ChatGPT appears to be the best ingestion strategy with an accuracy of 0.64 for only 1000 compact chunks.

Retrieval Strategies Evaluation

Table 3 and 4 show the retrieved chunk quality for SQuAD and COM datasets, respectively. 20 retrieval strategies including 4 HiRAG combinations are compared using BERT and ROUGE scores. The average number of tokens added to the prompt reflects the LLM inference latency.

Table 3: Retrieval strategies comparison on SQuAD

Retrieval strategy	BERT Score ↑	ROUGE- 1 ↑	2 ↑	L ↑	tok/ ctx ↓
FS (S)	0.82	0.14	0.06	.014	16
FS (M)	0.83	0.17	0.10	0.17	31
FS (L)	0.82	0.14	0.09	0.14	56
Rec. (S)	0.82	0.15	0.07	0.15	14
Rec. (M)	0.83	0.18	0.11	0.17	29
Rec. (L)	0.83	0.15	0.09	0.14	59
NLTK (S)	0.83	0.19	0.13	0.19	43
NLTK (M)	0.84	0.19	0.13	0.19	43
NLTK (L)	0.83	0.17	0.11	0.17	53
SpaCy (S)	0.84	0.19	0.13	0.19	43
SpaCy (M)	0.84	0.19	0.13	0.19	43
SpaCy (L)	0.83	0.17	0.11	0.17	53
HiRAG					
Llama2 QA chunks					
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{A}$	0.85	0.24	0.15	0.23	20
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{A}$	0.85	0.25	0.16	0.25	
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{QA}$	0.83	0.17	0.10	0.16	36
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{QA}$	0.83	0.18	0.11	0.18	
ChatGPT QA chunks					
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{A}$	**0.86**	**0.29**	**0.18**	0.28	**11**
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{A}$	**0.86**	**0.29**	**0.18**	**0.29**	
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{QA}$	0.83	0.19	0.10	0.18	28
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{QA}$	0.84	0.20	0.11	0.19	

Table 4: Retrieval strategies comparison on COM dataset

Retrieval strategy	BERT Score ↑	ROUGE- 1 ↑	2 ↑	L ↑	tok/ ctx ↓
FS (S)	0.86	0.28	0.15	0.25	16
FS (M)	0.87	0.36	0.25	0.33	31
FS (L)	0.88	0.35	0.26	0.32	56
Rec. (S)	0.87	0.31	0.19	0.28	**14**
Rec. (M)	0.88	0.38	0.28	0.36	29
Rec. (L)	0.88	0.35	0.25	0.32	54
NLTK (S)	0.89	0.41	0.30	0.38	26
NLTK (M)	0.89	0.42	**0.32**	0.39	30
NLTK (L)	0.88	0.38	0.29	0.36	51
SpaCy (S)	0.89	0.40	0.29	0.38	24
SpaCy (M)	0.89	0.41	0.31	0.39	29
SpaCy (L)	0.88	0.38	0.29	0.36	51
HiRAG					
Llama2 QA chunks					
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{A}$	**0.90**	0.42	0.31	0.39	24
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{A}$	**0.90**	**0.43**	**0.32**	**0.41**	
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{QA}$	0.88	0.35	0.24	0.32	39
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{QA}$	0.88	0.36	0.24	0.32	
ChatGPT QA chunks					
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{A}$	**0.90**	0.42	**0.32**	0.41	**14**
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{A}$	**0.90**	**0.43**	**0.32**	**0.41**	
$\text{Emb}_\text{Q} \wedge \text{Ret}_\text{QA}$	0.88	0.38	0.25	0.34	29
$\text{Emb}_\text{QA} \wedge \text{Ret}_\text{QA}$	0.88	0.38	0.25	0.34	

Table 3 shows uniform BERT scores across all retrieval strategies, indicating that the retrieved chunks are equally similar to the expected answer. In most cases, HiRAG obtains the best ROUGE scores, showing that the original knowledge is preserved. Specifically, Ret_A achieves higher scores than Ret_QA, as all metrics are influenced by chunk length; using only the answer results in shorter prompts compared to including the entire QA pair. On average, with only 11 tokens added to the prompt, the ChatGPT Ret_A configurations are the most successful retrieval strategies in this evaluation.

Table 4 exhibits a consistent trend, indicating that HiRAG achieves superior retrieval performance. HiRAG configurations implying Ret_A slightly improve over other strategies with BERT scores of 0.90. In particular, the ChatGPT $\text{Emb}_\text{QA} \wedge \text{Ret}_\text{A}$ configuration shows higher scores across all metrics. For the COM technical knowledge, the HiRAG solution improves the quality of retrieved chunks while limiting the latency introduced by adding context to the prompt.

Limitations

HiRAG database construction relies on computationally expensive LLMs to generate chunk databases. These LLMs can also be costly, contingent upon the chosen model and the volume of knowledge data. Furthermore, the generated QA pairs may require post-processing to ensure a consistent output format.

Our enhancement of the retrieved chunk quality may have a modest impact on LLM generation. While RAG consistently produces more accurate answers with fewer hallucinations, the differences between chunks from different strategies might become indistinguishable in terms of post-generation answer improvement.

Foundation LLMs are not trained to capitalize on the prompt format introduced by HiRAG chunks, highlighting an opportunity for further performance improvement. Although fine-tuning the LLM was beyond the scope of this study, training the model to leverage Ret_A chunks could enhance answer quality.

Conclusion

This paper presents HiRAG, a RAG alternative for embedded applications. Our experiments demonstrate that HiRAG outperforms other strategies, particularly when using ChatGPT and for specialized use cases. Compared to state-of-the-art chunking algorithms, HiRAG condenses data more efficiently and generates fewer chunks, while preserving the integrity of the original knowledge. Additionally, our study shows that QA chunk formatting maintains or improves chunk quality while reducing the number of added tokens to the prompt. Smaller embedding models, like GIST-all-MiniLM-L6-v2, benefit from the HiRAG format, leading to improved retrieval accuracy. Furthermore, the configuration that retrieves only the answer part of the chunks achieves the highest chunk quality with a reduced prompt, minimizing latency in embedded applications. Future work on fine-tuning the LLM to better leverage the HiRAG chunk format could enhance answer quality.

References

Abdin, M.; Jacobs, S. A.; Awan, A. A.; Aneja, J.; Awadallah, A.; Awadalla, H.; Bach, N.; Bahree, A.; Bakhtiari, A.; Behl, H.; et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint: 2404.14219*.

Bird, S. 2006. NLTK: The Natural Language Toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 69–72. Association for Computational Linguistics.

Boros, T.; Chivereanu, R.; Dumitrescu, S.; and Purcaru, O. 2024. Fine-Tuning and Retrieval Augmented Generation for Question Answering Using Affordable Large Language Models. 75–82. ELRA and ICCL.

Carraro, D.; and Bridge, D. 2024. Enhancing recommendation diversity by re-ranking with large language models. *ACM Transactions on Recommender Systems*.

Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; St. John, R.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Strope, B.; and Kurzweil, R. 2018. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics.

Dhar, N.; Deng, B.; Lo, D.; Wu, X.; Zhao, L.; and Suo, K. 2024. An empirical analysis and resource footprint study of deploying large language models on edge devices. In *Proceedings of the 2024 ACM Southeast Conference*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint: 2407.21783*.

Farouk, M. 2020. Measuring text similarity based on structure and word embedding. *Cognitive Systems Research*, 63: 1–10.

Gong, H.; Shen, Y.; Yu, D.; Chen, J.; and Yu, D. 2020. Recurrent Chunking Mechanisms for Long-Text Machine Reading Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6751–6761. Association for Computational Linguistics.

Hu, E. J.; yelong shen; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint: 2310.06825*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Association for Computational Linguistics.

Ma, X.; Fang, G.; and Wang, X. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36.

Muennighoff, N.; Tazi, N.; Magne, L.; and Reimers, N. 2022. MTEB: Massive text embedding benchmark. *arXiv preprint: 2210.07316*.

Neumann, M.; King, D.; Beltagy, I.; and Ammar, W. 2019. ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, 319–327. Association for Computational Linguistics.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2383–2392. Association for Computational Linguistics.

Reid, M.; Savinov, N.; Teplyashin, D.; Lepikhin, D.; Lillicrap, T.; Alayrac, J.-b.; Soricut, R.; Lazaridou, A.; Firat, O.; Schrittwieser, J.; et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint: 2403.05530*.

Solatorio, A. V. 2024. GISTEmbed: Guided In-sample Selection of Training Negatives for Text Embedding Fine-tuning. *arXiv preprint: 2402.16829*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint: 2307.09288*.

Venish, A.; and Siva Sankar, K. 2016. Study of Chunking Algorithm in Data Deduplication. In *Proceedings of the International Conference on Soft Computing Systems*, 13–20. Springer India.

Wang, L.; Yang, N.; Huang, X.; Yang, L.; Majumder, R.; and Wei, F. 2024. Improving Text Embeddings with Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 11897–11916. Association for Computational Linguistics.

Wang, W.; Wei, F.; Dong, L.; Bao, H.; Yang, N.; and Zhou, M. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33: 5776–5788.

Xiao, G.; Lin, J.; Seznec, M.; Wu, H.; Demouth, J.; and Han, S. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*. PMLR.

Ye, J.; Chen, X.; Xu, N.; Zu, C.; Shao, Z.; Liu, S.; Cui, Y.; Zhou, Z.; Gong, C.; Shen, Y.; et al. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *arXiv preprint: 2303.10420*.

Zhang, P.; Zeng, G.; Wang, T.; and Lu, W. 2024. Tinyllama: An open-source small language model. *arXiv preprint: 2401.02385*.

Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; and Artzi, Y. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.