International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Efficient Usage of RAG Systems in the World of LLMs

Priyank Jayantilal Rathod[1], Anurag[2]

[1,2]Intel Corporation, Folsom, CA

Abstract

The integration of Retrieval-Augmented Generation (RAG) systems with Large Language Models (LLMs) has revolutionized the field of Natural Language Processing (NLP). By leveraging RAG techniques, LLMs can access a broader range of information, improve coherence, and enhance the relevance of generated text. This paper explores the efficient usage of RAG systems in LLMs, highlighting their benefits, applications, and future implications.

Keywords: Retrieval-Augmented Generation (RAG), Embeddings, Large Language Models, Natural Language Processing.

Introduction

"*As of my last update in January 2022, I can't provide specific details about….*" or "*My knowledge has a cutoff date of January 2022…*" are common phrases that users of LLMs such as ChatGPT, developed by OpenAI, encounter when they need information on the latest events, developments, or technology from present time frames.

LLMs can reason about wide-ranging topics, but their knowledge is limited to the public data up to the specific time they were trained. Retrieval-augmented generation (RAG) is a cutting-edge technology that optimizes the output of language models by referencing an authoritative knowledge base outside of their training data sources. This approach enables the creation of more accurate, informative, and trustworthy responses. RAG involves augmenting user input with relevant retrieved data to enhance response generation. This approach leverages external knowledge sources to provide more accurate and up-to-date information, ensuring that language models generate responses that are grounded in reality. By combining the capabilities of language models with the power of search and retrieval, RAG enables the creation of more sophisticated AI applications. RAG systems contain three key modules: retriever, ranking, and generation. The retriever identifies relevant passages from a knowledge source based on context, while the ranking module sorts and prioritizes the retrieved passages. The generation module then uses this information to generate coherent and relevant text. Optimizing RAG for LLMs involves refining techniques across retrieval, ranking, and generation modules. This includes adapting network scale and architecture to improve the overall flow of the RAG system. RAG ensures that language models can access the most current and reliable facts, allowing them to generate more accurate responses.

Moreover, it provides users with transparent and trustworthy sources, enabling them to verify the authenticity of the generated responses. This approach has far-reaching implications for industries such as customer service, education, and healthcare, where accuracy and reliability are paramount.

Methodology

Introduced and pioneered by Lewis et al. in the paper[1], Retrieval-Augmented Generation (RAG) models are an approach that synergizes pre-trained parametric and non-parametric memory for enhanced language generation. It's a hybrid architecture that combines the strengths of pre-trained generator models with a retrieval component that accesses a dense vector index of external knowledge sources. This architecture allows RAG to incorporate relevant external knowledge into the generation process dynamically. Figure 1 gives a high-level architecture of RAG-based LLM applications. The model operates by first using a query encoder to retrieve top-K documents from the non-parametric memory based on the input query, and a generator then uses these documents as additional context to produce the final output(represented as Retrieval Tool and LLM, respectively in Figure 1).

RAG has two main components, working in tandem to leverage both parametric (learned during training) and non-parametric (external knowledge bases) memory:-

1. **Retrieval component**: The input query (e.g., a question or prompt) is first processed by a query encoder, typically a pre-trained transformer model. This encoder transforms the input query into a dense vector representation. Alongside the query encoder, RAG utilizes a dense vector index of documents from an external knowledge source. Each document in the index is represented by a dense vector, which is pre-computed using a document encoder (another transformer model) and stored for efficient retrieval. RAG performs a Maximum Inner Product Search between the query vector and the document vectors in the index to retrieve relevant documents based on the input query. This process identifies the top-K documents whose vector representations have the highest inner product (similarity) with the query vector. Act as information gatekeepers, searching through a large corpus of data to find relevant information for text generation.
2. **Generative component**: Synthesizes the retrieved information into coherent and contextually relevant text. The retrieved documents and the original input query are fed into a pre-trained model, which serves as the generator in the RAG architecture. The generator combines the context from the input query and the retrieved documents to generate a response informed by external knowledge.

RAG models can operate in RAG-Sequence and RAG-Token[1]. The same retrieved documents create the entire output sequence in RAG-Sequence. In RAG-Token, different documents can be used to make various output parts. RAG models are trained end-to-end, allowing both the retriever and generator components to be fine-tuned on downstream tasks. This training approach enables the retriever to learn which documents are most helpful in generating accurate responses, while the generator learns to incorporate the retrieved knowledge effectively. The training objective is typically the negative log-likelihood of the target sequence given the input and the retrieved documents. This objective encourages the model to generate coherent responses relevant to the input and factually accurate responses based on the retrieved knowledge.

Building Blocks for an efficient RAG system

Efficient prompts: Prompts or Prompt engineering is designing and crafting input prompts or queries to generative AI models to elicit desired outputs or responses. Research shows that mastering prompt engineering enhances the quality of information obtained from AI tools, thereby improving learning efficiency and task completion[2]. One of the challenges of working with LLMs is generating high-quality,

contextually appropriate responses with minimal data. Transforming input prompts into formats that are more easily understood by the models, such as using techniques like Query
Transformation Module (QTM) that refines input prompt sentences into more comprehensible forms for LLMs as proposed in this paper[3] makes it possible to achieve better performance from LLMs without the need for extensive additional training data or fine-tuning. The prompt's choice of words, format, and context can significantly influence the generated content. Be Clear and Specific, Specify the Format, Add Context, Use Examples, Control the Tone, Ask the Model to Think Step by Step, Use Keywords, Provide Constraints, Iterate and Refine.

Right Embedding Model: Retrieval-augmented generation is centered around embeddings[4]. Research suggests how significant word embeddings can be in improving the performance in practical information retrieval scenarios[5]. The MTEB[6] Leaderboard on Hugging Face[7] is a valuable resource for exploring the suitable embedding models to employ. The overall MTEB score gives a general idea of the top-performing embedding model, but it's best to sort the models by column on the leaderboard for the specific task one is interested in. Although benchmarks are a helpful starting point, remember that these outcomes are self-reported and may not fairly represent the data you are working with.

Given that the MTEB datasets are freely accessible, it is also probable that specific models will use them as part of their training set. It is best to test a model on your dataset even if one decides on it based on benchmark results.

State-of-the-art LLM: The evolution of large language models (LLMs) took a significant turn with the introduction of transformer models[8] in 2017. These models, characterized by their encoder-decoder architecture, revolutionized natural language processing (NLP) by enhancing the ability of machines to understand context and generate text. Transformers introduced two key innovations: word embeddings, which allowed models to grasp the meaning of words within their context, and attention mechanisms, which enabled models to determine the relevance of different words or phrases within a sentence. The first notable implementation of transformer technology was in models like GPT[9] (Generative Pre-trained Transformer) and BERT[10] (Bidirectional Encoder Representations from Transformers). GPT, with its ability to generate coherent and contextually relevant text, and BERT, focusing on understanding the context of words in a sentence, set new standards for what LLMs could achieve. The release of GPT-3 by OpenAI in 2020, with its 175 billion parameters, marked a watershed moment for LLMs. GPT-3's vast size and sophisticated architecture allowed it to perform a wide range of tasks with minimal task-specific training, from writing essays to generating code. The advent of transformer models has fundamentally changed the landscape of NLP by enabling a more nuanced understanding and generation of human language. This evolution has led to the development of increasingly powerful LLMs, culminating in state-of-the-art capabilities.

Factors to consider for RAG

- Retrieval Average: RAG is a retrieval task, so one should pick the best retrieval model for their application. Taking the MTEB Leaderboard as a reference and focusing on the Retrieval Average column, we can find the best scoring models that can be used for RAG. This column represents the average Normalized Discounted Cumulative Gain (NDCG) [11] across several datasets. NDCG is a standard metric for measuring retrieval systems' performance. A higher NDCG indicates a model that better ranks relevant items in the retrieved results list.

- Model Size: The model's size (usually in GB) is the memory footprint of the particular model. It gives an idea of the computational resources required to run the model. While retrieval performance scales with model size, it is essential to note that model size also directly impacts latency. The latency-performance trade-off becomes especially important in a production setup.
- Maximum Tokens: The fundamental data units that LLMs process are tokens. Based on the tokenization method, a token in the context of text can be a word, a subword, or a letter. Each model has a token limit, which is the number of tokens that an LLM can process in one transaction, and this limit can influence the performance of LLMs. Maximum tokens for RAG is the number of tokens compressed into a single embedding. As this paper[12] pointed out, LLMs exhibit a significant drop in reasoning performance as input length increases, even at lengths much shorter than their technical maximum.
- Embedding Dimensions: It is the length of the embedding vector. Smaller embeddings offer faster inference and are more storage-efficient, while more dimensions can capture nuanced details and relationships in the data. Ultimately, we want a good trade-off between capturing the complexity of data and operational efficiency.

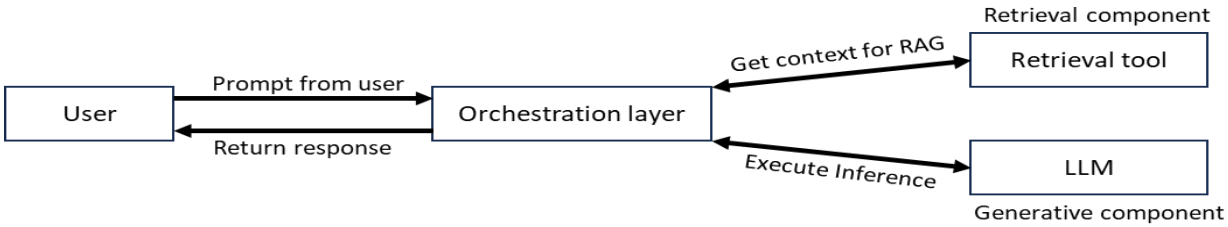

Figure 1. RAG-based application architecture

Advantages of RAG

- Enhanced Factual Accuracy and Specificity: RAG models generate more specific, diverse, and factually accurate responses than parametric-only seq2seq baselines. This is particularly evident in knowledge-intensive tasks such as open-domain question answering and fact verification, where RAG models have set new state-of-the-art benchmarks.
- Dynamic Knowledge Access: Unlike traditional models that rely solely on their parameters for knowledge, RAG can access and incorporate up-to-date information from external sources. This capability is crucial for tasks requiring current knowledge or specific details not contained within the model's pre-trained parameters.
- Interpretability and Provenance: RAG offers interpretability and provenance for its outputs by retrieving documents as part of the generation process. Users can inspect the retrieved documents to understand the basis of the model's responses, addressing a common critique of deep learning models as "black boxes."

Future Discussion

While RAG represents a significant advancement in LLMs, several research gaps and opportunities for future work remain:

- Retrieval Component Optimization: RAG's performance is contingent on the effectiveness of the retrieval component. Challenges such as retrieval collapse, where the model retrieves the same

documents regardless of the input, highlight the need for improved retrieval mechanisms and training strategies.

- Keyword-Based Retrieval: RAG's effectiveness is limited by its reliance on keyword searches, which may not capture the complexity of specific queries or the relevance of documents for abstract concepts.
- Scalability and Efficiency: The computational cost of retrieving documents from large external databases poses scalability challenges, especially for real-time applications. Future research could explore more efficient retrieval methods and indexing techniques to mitigate these challenges.
- Domain-Specific Applications: Investigating the application of RAG in domain-specific contexts, such as medical or legal NLP tasks, could reveal new challenges and opportunities for leveraging domain-specific knowledge bases.
- Handling Ambiguity and Uncertainty: RAG's reliance on external knowledge sources introduces the risk of propagating biases or inaccuracies present in these sources. Further research is needed to develop mechanisms for handling ambiguity, uncertainty, and potential biases in retrieved content.
- Hallucinations: While RAG can reduce the occurrence of hallucinations, it is not a comprehensive solution to the problem of AI models generating inaccurate information

Conclusion

In conclusion, Retrieval-Augmented Generation (RAG) emerges as a transformative approach in Large Language Models (LLMs), addressing the critical challenge of keeping AI-generated responses current and factually accurate. This technology leverages a multilateral system comprising retrieval, ranking, and generation modules to incorporate real-time data into the response generation process dynamically. The methodology underscores the synergy between pre-trained generator models and a retrieval component, facilitating the creation of responses that are not only contextually relevant but also grounded in factual accuracy. Efficient prompts, appropriate embedding models, and the selection of advanced LLMs are essential components to optimize RAG systems. The advantages obtained from RAG are multifold, such as enhanced factual accuracy, dynamic knowledge access, and improved interpretability, which collectively bolsters AI's trustworthiness and applicability in various sectors. This paper identifies pivotal areas for future research, including optimizing the retrieval component, exploring domain-specific applications, and developing strategies to address scalability, efficiency, and mitigating biases. The potential of RAG to reduce hallucinations in AI-generated content, while not a panacea, marks a significant step forward in the quest for more reliable and accurate AI systems**.** As the technology continues to evolve, it promises to revolutionize how we interact with and rely on AI for information retrieval.

References

1. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 793, 9459–9474
2. M. Wang, M. Wang, X. Xu, L. Yang, D. Cai and M. Yin, "Unleashing ChatGPT's Power: A Case Study on Optimizing Information Retrieval in Flipped Classrooms via Prompt Engineering," in *IEEE Transactions on Learning Technologies*, vol. 17, pp. 629-641, 2024, doi: 10.1109/TLT.2023.3324714.

3. D. Park, G. -t. An, C. Kamyod and C. G. Kim, "A Study on Performance Improvement of Prompt Engineering for Generative AI with a Large Language Model," in *Journal of Web Engineering*, vol. 22, no. 8, pp. 1187-1206, November 2023, doi: 10.13052/jwe1540-9589.2285
4. Jurafsky, Daniel; H. James, Martin (2000). Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition. Upper Saddle River, N.J.: Prentice Hall. ISBN 978-0-13-095069-7.
5. Galke, L., Saleh, A., & Scherp, A. (2017). Word embeddings for practical information retrieval. In M. Eibl, & M. Gaedke (Eds.), INFORMATIK 2017 (pp. 2155-2167). Bonn: Gesellschaft für Informatik. doi:10.18420/in2017_215.
6. Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.
7. https://huggingface.co/spaces/mteb/leaderboard.
8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, £., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 2017-Decem (Nips), 5999–6009
9. Radford, A., & Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training.
10. Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2019). BERT: Pretraining of deep bidirectional transformers for language understanding. NAACL HLT 2019 – 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies – Proceedings of the Conference, 1(Mlm), 4171–4186
11. K. J¨arvelin and J. Kek¨al¨ainen. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems (TOIS), 20(4):422–446, 2002
12. Levy, M., Jacoby, A., & Goldberg, Y. (2024). Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models. *ArXiv, abs/2402.14848*