Retrieval-Augmented Generation (RAG) Systems in Research and Information Management

**Abstract/Summary:** Recent advancements in Retrieval-Augmented Generation (RAG) systems represent a significant development in natural language processing, combining the generative capabilities of language models with external information retrieval mechanisms to enhance factual accuracy and contextual relevance. They offer a particularly valuable opportunity to develop tools with private and proprietary information or data that may not have been included in the model’s training corpus. We aim to develop a natural language query interface for data from the Library of Congress, integrating web scraping and data structuring techniques to enable intuitive and efficient information retrieval. By enhancing user interaction, increasing data accessibility, and streamlining information discovery, our research assistant enables scholars, investigators, journalists, and the general public to readily access the specialized and diverse information repositories housed at the Library of Congress.

**Introduction:** Retrieval-Augmented Generation (RAG) systems represent a significant advancement in the domain of natural language processing, particularly in enhancing the capabilities of large language models (LLMs). RAG operates by supplementing the generative process of LLMs with targeted information retrieval from external sources that extend beyond their training data, addressing their inherent limitations such as hallucination of facts and inability to update knowledge continuously. The indexing of this new information into a stored vector database enable the RAG system’s ability to identify relevant text that is similar to the user’s query.[[1]](#endnote-1)

The core components of RAG involve a retriever that locates and collects the necessary information from the vector database and a generator that incorporates this information to produce contextually rich and accurate responses. The retrieval component can operate using either dense or sparse vector spaces to ensure relevance and accuracy in the data retrieved, with the former capturing more complex semantic relationships between words and phrases that enable a more nuanced understanding and retrieval of information.[[2]](#endnote-2) ​ The generator is typically a fine-tuned version of an LLM that adapts its responses based on the information provided by the retriever​.[[3]](#endnote-3)

By enhancing this Naïve RAG system with additional pre-and-post retrieval steps…look at paper cited for the following: “Our investigation into Retrieval-Augmented Generation (RAG) techniques has identified HyDE and LLM reranking as notable enhancers of retrieval precision in LLMs. Surprisingly, established techniques like MMR and Cohere rerank did not demonstrate significant benefits, and Multi-query was found to be less effective than baseline Naive RAG. The results demonstrate the efficacy of the Sentence Window Retrieval technique in achieving high precision for retrieval tasks…[[4]](#endnote-4)”

* Sentence-window retrieval: In addition to retrieving the relevant sentences, Sentence Window Retrieval also includes the surrounding context – the sentences that come before and after the target sentence. This expanded context window is then fed into the language model for synthesis, ensuring that the generated answer has the necessary context for coherence and completeness. (easy reference at[[5]](#endnote-5))
* Hypothetical Document Embedding (HyDE): basically using the LLM to generate an answer to the query and then using the LLM’s answer to search the RAG system’s documents (easy reference at[[6]](#endnote-6))
* LLM Reranking: retrieved documents are further assessed for relevance using a reranking step, which refines the set of documents that will inform the generated response
* Explore if time permitted: Integrating Knowledge Graphs (KGs) with RAG systems represents a promising direction for enhancing retrieval precision and contextual relevance.

Implementing a RAG system as a research assistant for the Library of Congress could enhance the way information is accessed and utilized, offering a powerful tool for researchers that can accelerate the research process. By dynamically integrating the vast repository of documented knowledge with state-of-the-art language models, a RAG system can provide precise, contextual, and up-to-date information. This would not only enhance the factual accuracy of the responses but also significantly reduce the time researchers spend navigating through extensive archival data. Moreover, the flexibility of RAG to adapt to different types of queries and its ability to handle complex information requests can make it an invaluable asset in managing large-scale information repositories like those of the Library of Congress​.[[7]](#endnote-7)

**RAG System:** Data…

After parsing and processing the data, our documents are broken up into chunks of 100 characters using Langchain’s RecursiveCharacterTextSplitter. A 15% overlap between the chunks is applied to maintain context and help improve retrieval accuracy. We used Instructor-XL as the default embeddings model (or MiniLM) and Deeplake vectorstore to house the embedded data.

The top three documents are retrieved based on a user query and the embeddings and combined into a single string; the metadata for the best match is separately saved. A prompt is constructed with detailed instructions, the user’s query, combined documents, and the best document’s metadata. Based on the prompt, Meta’s Llama3.2-3b generates a response to the query by referencing the information it is supplied with. Given the purpose of the system as a research assistant, the generated output is constrained by a temperature of 0.7 to reduce the risk of hallucinations and ensure the text’s integrity. Finally, the metadata from the best matched document that was retrieved is added to the response to serve as a reference and citation for the generated text so that the user can easily refer to the original document if needed.

Evaluation…

Llama3 tokenizer go to 1064? - <https://mychen76.medium.com/multi-gpu-training-for-llama-3-2-using-deepspeed-and-redundancy-optimizer-zero-96a655ee63a8>

* Examples of good prompting for llama: <https://www.llama.com/docs/how-to-guides/prompting>

1. Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." *arXiv preprint arXiv:2312.10997* (2023). <https://arxiv.org/abs/2312.10997> [↑](#endnote-ref-1)
2. Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The Power of Noise: Redefining Retrieval for RAG Systems. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24). Association for Computing Machinery, New York, NY, USA, 719–729. <https://dl.acm.org/doi/abs/10.1145/3626772.3657834> [↑](#endnote-ref-2)
3. Wu, S., Xiong, Y., Cui, Y., Wu, H., Chen, C., Yuan, Y., ... & Xue, C. J. (2024). Retrieval-Augmented Generation for Natural Language Processing: A Survey. arXiv preprint arXiv:2407.13193. <https://arxiv.org/abs/2407.13193> [↑](#endnote-ref-3)
4. Eibich, Matouš, Shivay Nagpal, and Alexander Fred-Ojala. "ARAGOG: Advanced RAG Output Grading." *arXiv preprint arXiv:2404.01037* (2024). <https://arxiv.org/pdf/2404.01037> [↑](#endnote-ref-4)
5. <https://www.linkedin.com/pulse/sentence-window-retrieval-optimizing-llm-performance-rutam-bhagat-v24of> [↑](#endnote-ref-5)
6. <https://jayant017.medium.com/rag-using-langchain-part-5-hypothetical-document-embeddings-hyde-050f57dfc252> [↑](#endnote-ref-6)
7. Hu, Yucheng, and Yuxing Lu. "Rag and rau: A survey on retrieval-augmented language model in natural language processing." arXiv preprint arXiv:2404.19543 (2024). <https://arxiv.org/abs/2404.19543> [↑](#endnote-ref-7)