A diagram of a process

Description automatically generated

Retrieval-Augmented Generation (RAG), proposed by Lewis in mid-2020, refers to the process where Large Language Models (LLMs) retrieves relevant information from data source before generating text for response. RAG combines retrieval and In-Context Learning (ICL) techniques to improve the quality of response. When a user prompt is inputted, relevant information is retrieved from external knowledge bases using search algorithms and provided as context by incorporating the contextual information into the user’s prompts. This eliminates the need for developers to retrain the model for specific-domain tasks while improving the accuracy of the responses.

|  |  |  |  |
| --- | --- | --- | --- |
| **RAG Approach** | **Description** | **Strength** | **Weakness** |
| Naïve RAG | **Indexing**  Involves cleaning and extracting data from various formats into plain text. Then, large documents are divided into smaller chunks because LLM has a limited context window. The text is encoded into vectors using a language model.  **Retrieve**  Select top K relevant documents from knowledge bases based on level of similarity (can be calculated using cosine similarity).  **Generation**  User prompts and the retrieved document chunks are concatenated into a new prompt. Then, a response is generated based on the new prompt. | * **Cost effective**: Compared to fine-tuning the model, RAG requires less computational resources as it does not need to re-train the model. * **Performance**: Outperforms the native LLM without RAG because it incorporates contextual information. | * **Quality of retrieval**: Low precision because the retrieve chunks do not always align well with the query, leading to hallucination. * **Quality of response**: Model generates an answer that is not rooted in the provided context (hallucination). The model sometimes also gives irrelevant response to user prompts. * Retrieved content may originate from various writing styles or tones, so ensuring consistency in the output is a challenge. |
| Advanced RAG | In addition to the processes in Naïve RAG, Advanced RAG introduces **pre-retrieval and post-retrieval strategies**. The indexing approach is refined using techniques like sliding window, fine-grained segmentation and metadata. | * Another layer of context is appended via domain-specific annotations which can be enhanced with continuous updates through user feedback loops to **maintain the context** in real-world scenario. * **Outdated information is removed or updated** automatically. * **Increased indexing speed** with metadata information * Combines the strength of various search techniques for **better retrieval quality**. | * Complex architecture and implementation, which means more development and testing time is required. * More resources are needed compared to Naïve RAG. |
| Modular RAG | * **Search Modules**: Search any type of data sources aside from textual data. * **Memory Module**: Identify memories most similar to current input to guide retrieval. * **Extra Generation Module**: Uses LLM to generate necessary context based on the retrieve content so it is more likely to contain pertinent information. * **Alignment Module**: Trainable Adapter module to effectively mitigate alignment issue driven by LLM rewards (reinforcement learning). * **Validation Module**: Introduced to assess the relevance between the retrieved documents and the query, increasing robustness of RAG. | * More varieties of external knowledge can be stored and retrieved when needed. * Increased flexibility and adaptability as it allows substitution or rearrangement based on specific needs or issues. | * Complex data integration as it involves many types of data format. * Involves routing scheduling, and decision-making processes that require precise control over how each module interacts, increasing the complexity in designing an effective orchestration module. * Requires careful consideration, design and continuous optimization to ensure all components are well-integrated. |

References:

* <https://blog.jayanthk.in/types-of-rag-an-overview-0e2b3ed71b82>
* <https://export.arxiv.org/pdf/2312.10997v2>
* <https://newsletter.nocode.ai/p/comprehensive-guide-rag-implementations> [specific types of RAG]
* <https://medium.com/@sahin.samia/modular-rag-using-llms-what-is-it-and-how-does-it-work-d482ebb3d372> [Modular RAG]
* <https://www.smashingmagazine.com/2024/01/guide-retrieval-augmented-generation-language-models/> [Implementation of RAG for Llama]

Tools Used:

* Hugging Face
* PhiData
* LangChain