Paper ML06

Unlocking the potential of your data with Retrieval Augmented Generation (RAG) while upholding privacy: use cases with business impact at Roche.

Adam Forys, Hoffmann-La Roche, Basel, Switzerland
Mathieu Cayssol, Hoffmann-La Roche, Basel, Switzerland

ABSTRACT

The launch of ChatGPT in November 2022 ignited a transformative revolution across all the industries, including the pharmaceutical industry, to enhance innovation and productivity. One of the common Large Language Model use cases is the development of Retrieval Augmented Generation (RAG) solutions built on specific knowledge bases. It consists of a ChatGPT-like application that utilizes enterprise knowledge, upholding data privacy, and operates cost-effectively. Through this revolution, many individuals often find themselves overwhelmed by the complexities of integrating advanced AI technologies into their workflows.

In our presentation, we delve into the world of Large Language Models (LLMs), focusing on RAG architectures. We break down the tech stacks and infrastructure needed to create these applications and emphasize the importance of data privacy considerations throughout the development process. In addition, we are sharing some exciting projects developed internally at Roche that are boosting efficiency. This concise overview of RAG architecture and concrete technologies, aiming to guide you in successfully crafting LLM-based products.

INTRODUCTION

Large Language Models have demonstrated outstanding capabilities to do human-like tasks such as generating content, summarizing text, translation, answering questions and generating code. Nevertheless, they have some drawbacks, including the potential for generating inaccurate or biased information and struggling with tasks requiring up-to-date knowledge. The Retrieval-Augmented Generation (RAG) architecture mitigates these drawbacks. It combines the strengths of LLMs with a retrieval mechanism to access and incorporate relevant, accurate information from external sources, thereby enhancing the reliability and accuracy of the generated content. The paper will present a brief overview of LLMs and the RAG Architecture, then presenting 2 internal use cases: OCEAN Assistant and the code completion copilot.

LLMs

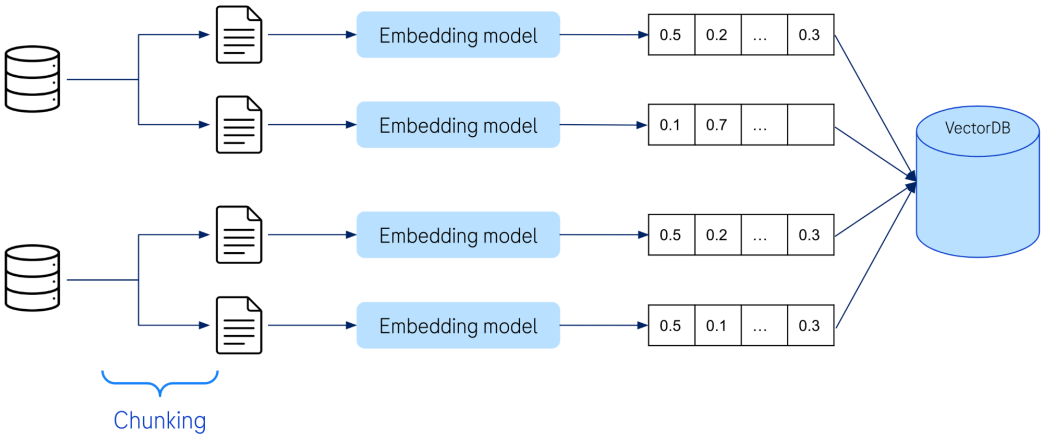
Large language models are advanced artificial intelligence systems that utilize deep learning techniques, with the most common being transformer architectures, to process and generate human-like text. They are trained on massive datasets to understand context, language patterns, and generate coherent and contextually relevant responses.

EMBEDDING AND SIMILARITY SEARCH

Embeddings are numerical representations of text that capture the meaning and context of words, sentences, or documents. These vector-based representations help machines understand relationships between different pieces of text. A common use case is similarity search, which leverages these embeddings to find related information. For

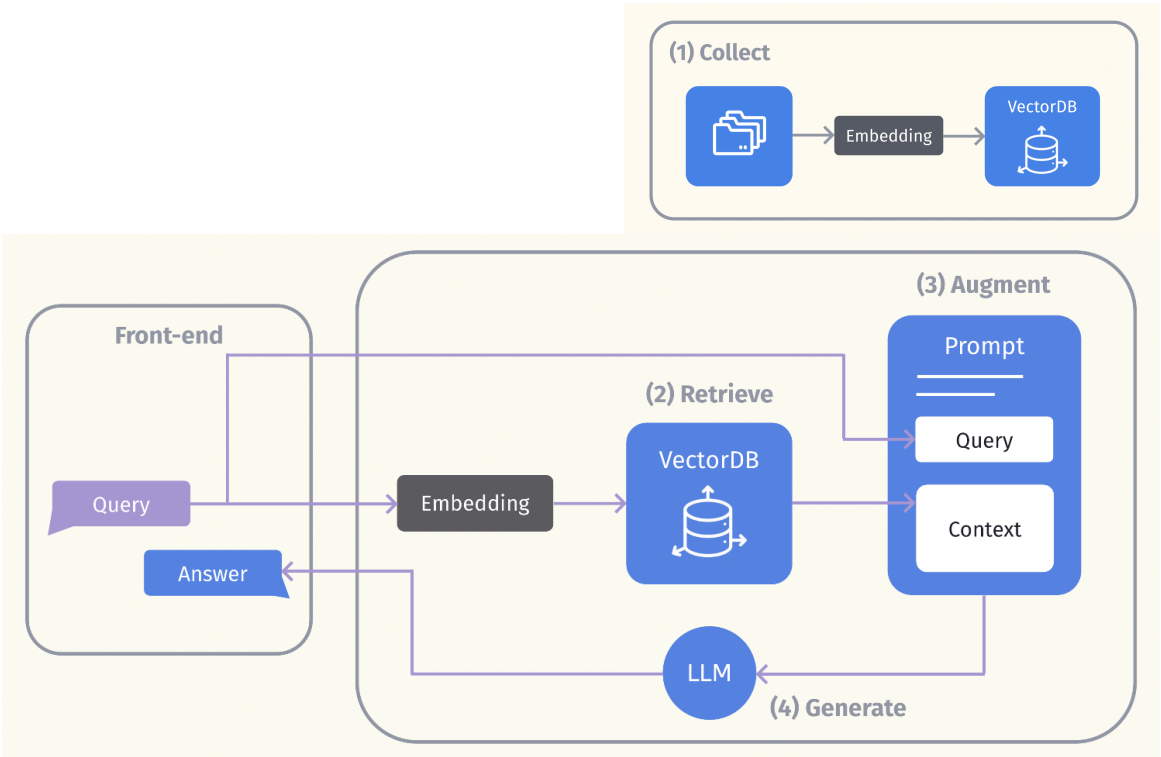
example, given a query, a similarity search can identify the most relevant pieces of content within an organization's knowledge base.

The diagram below illustrates how embeddings are generated from documents and stored in a Vector Database (VectorDB) to enable similarity search. By breaking documents into smaller chunks and processing them through an embedding model, the resulting vectors are stored for efficient retrieval.



RETRIEVAL AUGMENTED GENERATION (RAG)

Retrieval Augmented Generation (RAG) combines traditional language models with a retriever module to enhance content generation. It leverages pre-existing knowledge by retrieving relevant information from external sources, allowing the model to generate more informed and contextually rich responses. RAG is particularly useful for tasks that require incorporating external information, such as question answering or content creation.



The architecture can be broken down into 4 steps:

1. **Building the knowledge base**
   Identify Data Sources: Determine the sources of data to be used and extract textual information.

   Perform chunking: split the large text corpus into smaller, manageable pieces or segments. Each chunk acts as a standalone unit of information that can be individually indexed and retrieved.

   Generate Embeddings: Transform each chunk of text into a vector. The operation is called embedding and consists in storing the semantic meaning of the chunks into a numerical format using an embedding model.

   Store in the VectorDB: Save each vector into the vectorDB and add metadata associated with each vector.

2. **Retrieve**
   The retrieval process involves generating an embedding from the user's input and retrieving the most similar vector(s) from the vectorDB. To find the most similar vectors, a similarity metric is required. The cosine similarity is the most popular one. It measures the cosine of the angle between two vectors, thereby quantifying their similarity.

3. **Augment**
   The Augmentation step consists in crafting a prompt containing the instruction for the LLM, the query from the user and the retrieved chunks.

4. **Generate**
   The Generate step consists in providing the augmented prompt (instruction + question + top-k retrieved chunks) to the LLM.

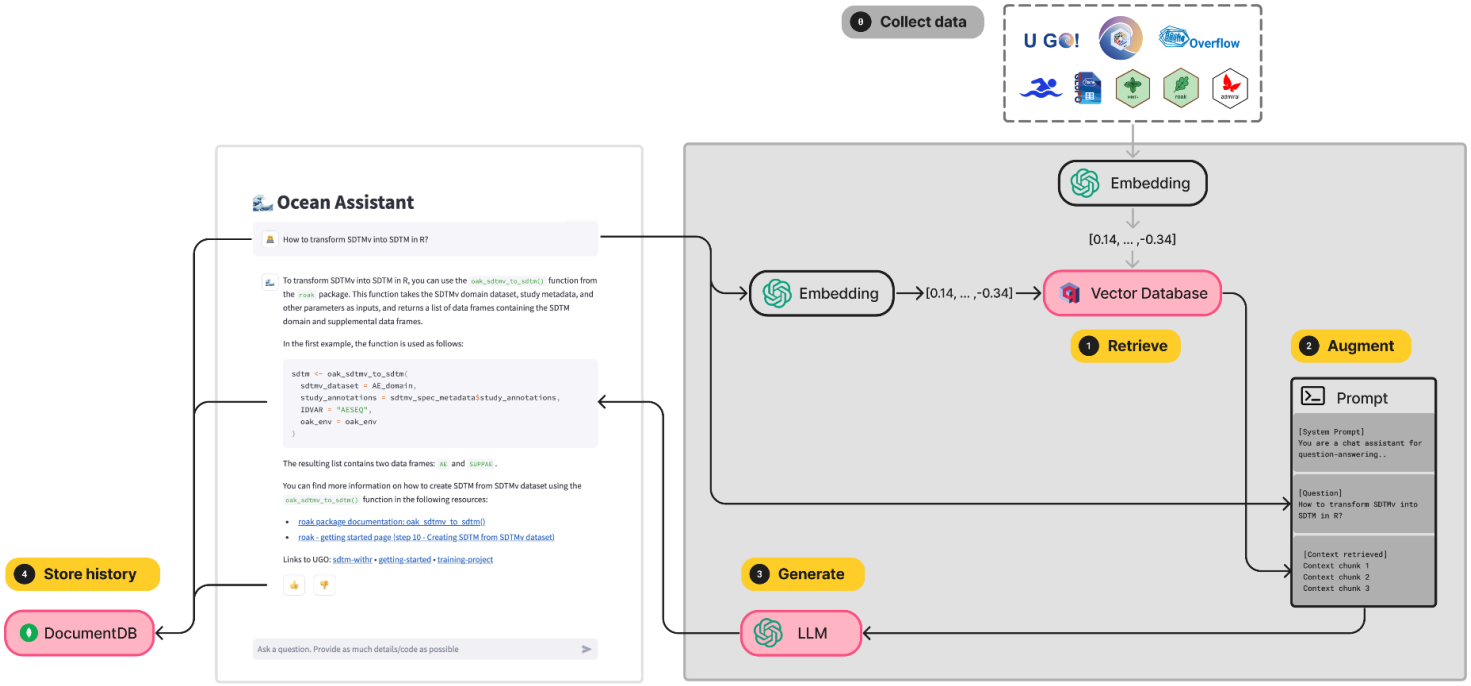
DATA CONTROL WITH RAG

RAG (Retrieval-Augmented Generation) enhances security and data privacy by keeping proprietary data within a secured database environment. This allows for strict access control, as the data is only accessed when needed and remains isolated from the model during training. In contrast, fine-tuning integrates your data into the model's training set, potentially exposing it to broader access and reducing visibility and control. By using RAG, organizations can ensure sensitive information is accessed only under controlled conditions, minimizing the risk of unauthorized access and data breaches.

OCEAN ASSISTANT: A CHATBOT BUILT ON TOP OF A KNOWLEDGE BASE

Business Context: OCEAN (One CEntralized ANalytics) is the next generation platform for the statistical programmers to deliver studies. The transition to this new platform is accompanied by a shift to a multi-paradigm ecosystem with commercial software (SAS ®) and open source software (R, Julia, Python). Despite the extensive documentation and training materials made available, the learning curve remains steep and manual support needs to be provided.

In order to accelerate the training of our statistical programmers and automate the manual support, we have created a chatbot to help users navigate through the new platform, tools and processes. OCEAN Assistant is a RAG application for question/answer tasks built on top of OCEAN's related documentation. It is designed to help users find answers to their questions quickly and efficiently. More than fifteen data sources are part of the OCEAN Assistant knowledge base and they are updated every week. The architecture is a typical RAG application.



The main capabilities are the followings:

- Summarize content from the documentation
- Retrieve documentation link
- Generate simple snippet of code
- Solve potential errors you encounter
- Explain processes
- Retrieve forum discussions

The OCEAN assistant has answered more than +23,000 questions, with ~250 questions per day. We have ~65% of positive feedback on the answer provided by the chatbot. The average number of questions per session is 2.5. In the future, we are planning to leverage the LLM-as-a-Judge paper approach for evaluating the quality of the answers.

CODE COMPLETIONS TOOL: IN-HOUSE COPILOT TO PROGRAM FASTER

Business Context: *completions* R package is an internal project aimed at improving developer productivity through intelligent code completion. This tool leverages LLMs to suggest code snippets based on the context, significantly reducing the time needed to write complex functions.

While there are commercial products available on the market, they do not yet offer the same level of flexibility as an internal solution. By leveraging metadata, you gain precise control over which code version is used, ensuring that package code remains separate from user-generated code. With proper data processing, you can extract only the information needed for specific tasks, thereby optimizing efficiency. Additionally, you retain the ability to seamlessly migrate between different LLM models, ranging from commercial solutions to fully open-source alternatives.

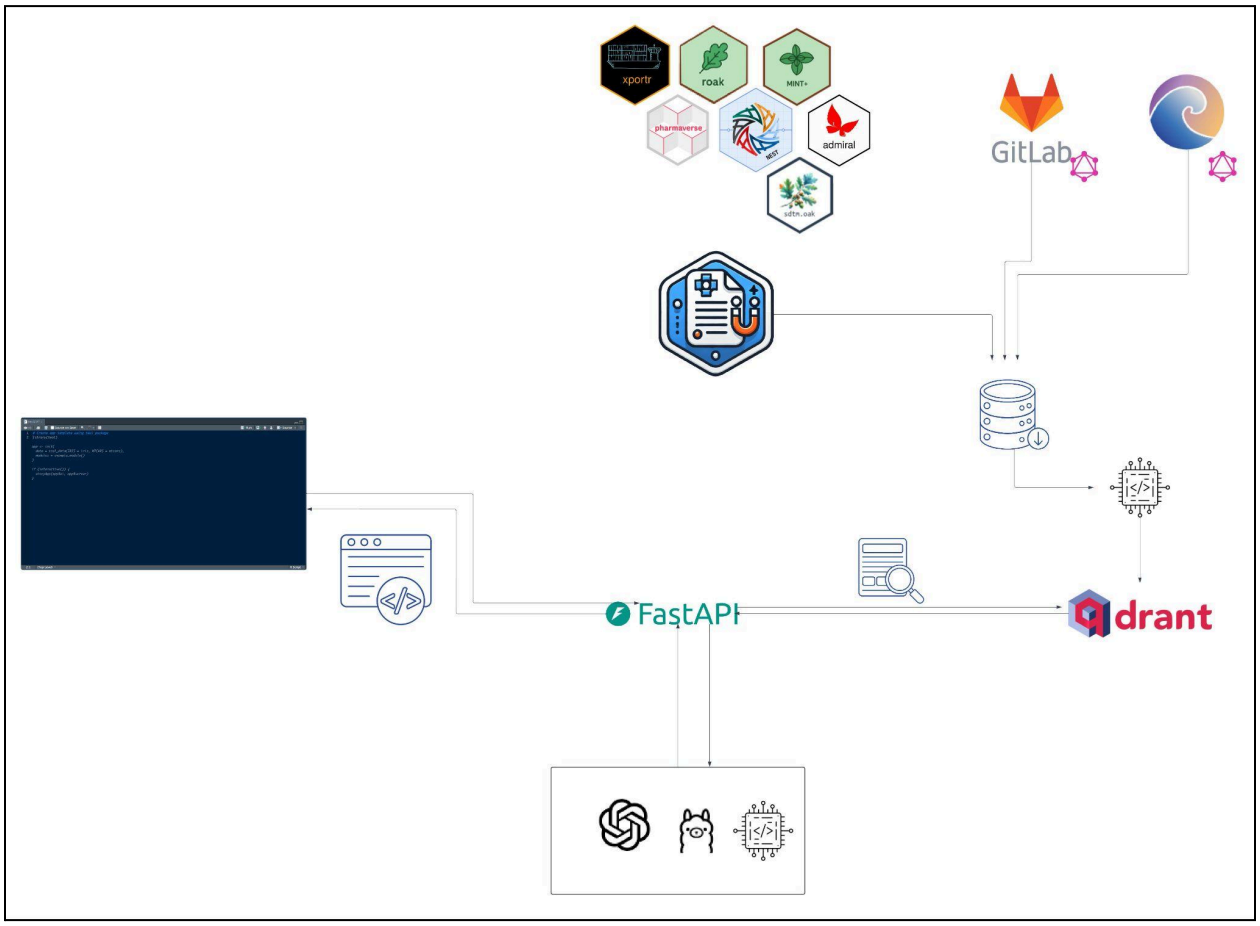
One major advantage of preprocessing code is the ability to merge data with OCEAN metadata. Since the OCEAN platform adopts FAIR data principles, we can easily locate and filter code repositories that meet our quality standards.

Information from the code repository is directly linked to package versions, generated outputs, and pipeline statuses. With all this metadata, maintaining the knowledge corpus becomes much easier.

We have decided to use the Fill-in-the-Middle (FIM) method, which is commonly used in training large language models to generate or complete text by filling in missing sections within a sequence. Unlike traditional left-to-right (autoregressive) generation, FIM allows the model to predict or generate text in the middle of a given prefix and suffix. This approach enhances the flexibility of language models, making them particularly effective for tasks involving editing, infilling, or completing code or text where missing sections need to be generated based on the surrounding context.

For the code completion API, we chose the DeepSeek Coder V2 model because it is both fast and cost-efficient. At any point, we have the flexibility to switch to a different model, whether a self-hosted solution or a commercial API, thanks to the use of RAG.

The new API functions introduced in the RStudio IDE enable the creation of custom package extensions that can manipulate documents. This update allows developers to build custom handlers in RStudio, which can provide code completions within document content.



CONCLUSION

The integration of LLMs, specifically through RAG architectures, presents a significant opportunity for enterprises to enhance productivity, innovation, and decision-making capabilities. The projects at Roche, such as the Ocean Assistant and code completions, demonstrate how LLM-based solutions can be developed to meet specific organizational needs while maintaining data privacy and ensuring cost-effectiveness. This paper provides a foundation for understanding the practical aspects of implementing RAG solutions, guiding readers through the steps needed to create impactful AI-driven applications.

ACKNOWLEDGMENTS

The work presented in this paper is the result of the collaborative efforts of R community, pharmaverse, studies teams and OCEAN team that provide the knowledge base for RAG.

CONTACT INFORMATION

Contact the authors at:
mathieu.cayssol@roche.com, adam.forys@roche.com

REFERENCE

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need.
arXiv:1706.03762 **[cs.CL]**
[2] Video explaining the 'Attention is all you need' architecture.
https://www.youtube.com/watch?v=4Bdc55j80l8
[3] Hugging Face. The GitHub for deep learning/machine learning models.
https://huggingface.co/models
[4] IBM. What is a RAG explained.
https://www.ibm.com/architectures/hybrid/genai-rag
[5] Video explaining how to improve your RAG.
https://www.youtube.com/watch?v=TRjq7t2Ms5I
[6] Qdrant Official documentation - vector Database
https://qdrant.tech/documentation/overview/
[7] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, Ion Stoica (2023).
Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena
arXiv:2306.05685 **[cs.CL]**
[8] Horizon-Length Prediction: Advancing Fill-in-the-Middle Capabilities for Code Generation with Lookahead Planning arXiv:2410.03103
[9] DeepSeek-Coder-V2: Breaking the Barrier of Closed-Source Models in Code Intelligence arXiv:2406.11931