Rag and ReAct from scratch

Nicolas Fidel
SCIA
EPITA Paris
14-16 Rue Voltaire, 94270
nicolas.fidel@epita.fr

Quentin Fisch
SCIA
EPITA Paris
14-16 Rue Voltaire, 94270
quentin.fisch@epita.fr

Adrien Giget
SCIA
EPITA Paris
14-16 Rue Voltaire, 94270
adrien.giget@epita.fr

Théo Ripoll
SCIA
EPITA Paris
14-16 Rue Voltaire, 94270
theo.ripoll@epita.fr

Abstract

In the rapidly evolving field of artificial intelligence, the integration of Retrieval-Augmented Generation (RAG) models into practical applications represents a frontier in enhancing machine understanding and response capabilities. This project explores the development of a RAG model specifically tailored for processing and understanding PDF documents, leveraging the ReAct framework. Initially, our work involved utilising Langchain to establish a foundational understanding of the potential efficacy of a ReAct-based RAG agent. Through this preliminary phase, we demonstrated promising results, as detailed in our langchain_react_rag.ipynb notebook.

Then we developed a method for extracting and chunking text from PDF documents, followed by computing embeddings for storage in a vector database : Chroma, a critical step towards enabling efficient retrieval and generation processes.

Here we show the successful implementation of a RAG model that operates without reliance on Langchain, marking a significant advancement in our project's objective to create an autonomous agent from scratch. This achievement not only showcases the versatility and potential of combining RAG models with the ReAct framework but also highlights the importance of prompt engineering in optimising model performance.

Our work contribute to a deeper understanding of the capabilities and limitations of RAG models in processing complex document formats. Everything can be found on our GitHub repository : `https://github.com/QuentinFISCH/wikipedia-react-rag`.

1 Introduction

1.1 Background

Retrieval-Augmented Generation (RAG) presented by [Lewis et al., 2021] represents a transformative approach in artificial intelligence, merging the generative capabilities of large language models (LLMs) with the precision of fact-based retrieval systems. This framework significantly enhances the accuracy and reliability of generated content by grounding responses in external, verifiable information. By doing so, RAG models address a critical limitation of standalone LLMs—their tendency to generate plausible but inaccurate or outdated information, a challenge particularly pronounced in rapidly evolving knowledge domains.

The ReAct framework draws inspiration from the inherent human ability to blend "acting" and "reasoning". This synergy allows for the learning of new tasks and making informed decisions through a combination of cognitive processes and interactions with the environment. ReAct aims to embody this principle in AI systems, facilitating a more dynamic, adaptive approach to learning and problem-solving that goes beyond static knowledge bases. This concept is thoroughly explored and presented in the work by [Yao et al., 2023], where they introduce "ReAct: Synergizing Reasoning and Acting in Language Models" as a novel framework designed to enhance the capabilities of AI systems by integrating the intricacies of human-like reasoning with the flexibility of action-based learning. This approach signifies a significant step towards creating AI that more closely mimics the depth and adaptability of human cognition and behavior

Chain-of-thought (CoT) prompting has emerged as a powerful technique to elucidate the reasoning processes of LLMs, enabling them to tackle complex tasks like arithmetic and commonsense reasoning with more transparency and coherence. Introduced by [Wei et al., 2023], CoT prompting encourages LLMs to generate intermediate steps or "thoughts" that lead to a final answer, mimicking human-like reasoning patterns. However, despite its strengths, CoT prompting does not inherently solve the issue of LLMs' limited access to real-time information or their inability to update their knowledge base, leading to potential inaccuracies or "fact hallucinations."

1.2 Motivation

The integration of RAG models within the ReAct framework for processing PDF documents is motivated by the need to enhance the accuracy and dynamism of document understanding technologies. PDFs, as a prevalent format for disseminating complex information across various fields, pose unique challenges for AI due to their diverse layouts, embedded images, and often non-linear text flow. Traditional LLMs, even when equipped with CoT capabilities, struggle to effectively parse and interpret the nuanced, dense information contained within PDF documents without access to updated, external knowledge bases or the ability to reason through content dynamically.

By leveraging RAG within the ReAct framework, this project aims to bridge the gap between static document content and the evolving external knowledge landscape. This approach not only seeks to mitigate issues like fact hallucination and error propagation inherent in LLMs but also to harness the potential of AI to understand and interact with complex document formats in a more human-like, adaptive manner.

1.3 Contributions

- Adrien Giget mainly worked on the underlying LLM model choice and prompt engineering, trying to find the best way to have answer according to the ReAct framework.
- Nicolas Fidel worked on the vector database choice and setup, as well as the development of the main tool.
- Théo Ripoll focused on the creation of the PDF loader and storage in the vector database, as well as the setup of the latter.
- Quentin Fisch worked initially on a Langchain exploration to see potential results, further focused on the development of the ReAct framework tool as well as some prompt engineering research.

1.4 Project presentation

In the culmination of our project, we successfully developed RAG model tailored for the comprehension and processing of PDF documents, utilising the ReAct framework. Initially envisioned to explore the intersection of advanced AI techniques and document understanding, the project's trajectory refined our goals to focus on creating a model capable of navigating the complexities of PDF content for efficient information retrieval and knowledge synthesis.

The motivation behind this initiative was to address the challenge of extracting and leveraging the wealth of information locked within PDF documents, a common but often underutilised data source due to its structured complexity. Recognising the potential to enhance accessibility and usability of this information, we embarked on developing a solution that bridges the gap between static document content and dynamic AI-driven analysis.

Our project's real-world applications span various sectors where PDF documents are prevalent. In academia, it can improve the way researchers interact with scientific literature, enabling swift reviews and information extraction. Legal professionals could use the model to expedite case document analysis, while industries reliant on technical manuals and reports, such as engineering and healthcare, could benefit from improved access to critical information, thus enhancing decision-making processes and operational efficiency.

1.5 Achievements

- Initial exploration and understanding of the ReAct framework's potential through the use of the Langchain library, establishing a foundational benchmark for the project's objectives.
- Selection and successful integration of Llama2 as the project's Large Language Model (LLM), chosen for its ease of use, performance, and compatibility with open-source tools, particularly through Ollama for local model execution.
- Development and implementation of a method for extracting and chunking text from PDF documents, enabling the processing of complex document formats into manageable and analysable data segments.
- Successful embedding of text chunks and storage in ChromaDB, a vector database, facilitating efficient retrieval of information necessary for the RAG model's operation.
- Achievement of an autonomous Retrieval-Augmented Generation (RAG) model operation, independent of Langchain, tailored for the processing and understanding of PDF documents within the ReAct framework.
- Overcoming challenges in prompt engineering and system prompts, refining the model to produce parse-able outputs in diverse situations, ensuring the RAG model's outputs could be effectively utilised within the ReAct framework.
- Demonstrating the model's potential through qualitative tests, showcasing its ability to handle the complexities of PDF document processing and information retrieval, aligning with the project's goals.

2 Related works

2.1 Research

As already mentioned, our work is possible by the implementation of RAG technique drawing from the foundational principles laid out in recent advancements. Particularly, [Lewis et al., 2021] have pioneered the Retrieval-Augmented Generation (RAG) technique, demonstrating its efficacy in enhancing the generative capabilities of language models through the incorporation of external knowledge bases. This approach has been instrumental in our research, enabling our models to access and leverage vast amounts of information beyond their initial training data. Similarly, [Yao et al., 2023] introduces a novel paradigm where language models synergize reasoning and acting, highlighting the importance of integrating dynamic decision-making within the computational framework. Furthermore, the Chain-of-Thought prompting method, proposed by [Wei et al., 2023], has shown significant promise in eliciting sophisticated reasoning processes in large language models. By guiding models through a series of logical steps, this method facilitates the generation of more coherent and contextually relevant responses. Collectively, these works form the cornerstone of our approach, underscoring the critical role of retrieval mechanisms and structured reasoning in advancing the capabilities of language models.

Combining those approaches has already been seen in the literature, reflecting a growing trend towards integrating sophisticated language models with domain-specific challenges. For instance, at NVIDIA Tsai et al. [2024]'s RTLFixer framework leverages the RAG technique alongside ReAct prompting, manifesting a significant stride in employing LLMs for debugging and fixing syntax errors in Verilog code. This convergence of methodologies not only underscores the versatility of

LLMs in addressing intricate problems but also exemplifies the practical applications of merging retrieval-based learning with interactive reasoning.

Similarly, the exploration by [He and Yu, 2024] into the potential of LLMs in transforming EDA processes—from design and verification to optimisation—further illustrates the potential synergy between advanced computational models and specialised engineering disciplines. Their analysis provides a critical perspective on the role of LLMs in EDA, proposing a future where these models could significantly impact efficiency, accuracy, and innovation in design automation.

These examples highlight a broader trend of leveraging LLMs to solve complex, domain-specific problems, reflecting an evolving landscape where the boundaries of what can be achieved with artificial intelligence are continually expanding.

2.2 Industrial Applications

By Google in December 2023, blog post : `https://cloud.google.com/blog/products/databases/introducing-sample-genai-databases-retrieval-app?hl=en` and related GitHub : `https://github.com/GoogleCloudPlatform/genai-databases-retrieval-app`.

- This was one of our inspirations as they use both Rag and ReAct.
- Create a demo of an AI assistant based at San Francisco airport, with access to information about airports, flights and amenities.
- The demo as it is forces you to use Google Cloud services.

By AWS, they have included the Rag and ReAct approach in their documentation: `https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/develop-advanced-generative-ai-chatbots-by-using-rag-and-react-prompting.html`, with the associated GitHub : `https://github.com/awslabs/genai-bedrock-chatbot`.

- Prerequisites AWS tools such as AWS CDK Toolkit 2.114.1+ configured, an active AWS account bootstrapped using AWS CDK, and access to the Bedrock service.
- Knowledge of AWS is a must for proper use.

The combination of both techniques is used in several **Medium articles**:

- By Dr Varshita Sher at Haleon: `https://towardsdatascience.com/using-langchain-react-agents-for-answering-multi-hop-questions-in-rag-systems-893208c1847` with a Jupyter Notebook demo : `https://github.com/V-Sher/LangChain_ReAct_Demo`.
- By Wenqi Glantz at ArisGlobal : `https://betterprogramming.pub/exploring-react-agent-for-better-prompting-in-rag-pipeline-b231aae0ca7c` that rely on AWS Snowflake : `https://github.com/wenqiglantz/react-agent-amazon-exhibit99.1`.

And even though, as we show in 3.1 with our langchain experiment, that it's very easy to have a working Rag + ReAct model, there's no one-step method for doing so.

3 Project

This project is not the easiest to explained in very distinct sections, so this part will describe in a chronological order how the project evolved.

3.1 Langchain utilisation in Exploring the ReAct Framework

The initial phase of our investigation may initially appear counter intuitive; however, it was imperative for delineating the trajectory and scope of our research. We employed the Langchain library, a pivotal step aimed at elucidating the operational dynamics and potential outcomes of the ReAct framework within our project's context. Langchain's reputation for delivering robust and compelling results precedes it, setting a high benchmark for our endeavours. Such efficacy nearly challenges the impetus for further exploration, given the ostensibly low likelihood of surpassing Langchain's performance benchmarks.

This section presents an overview of our preliminary exploration into the ReAct RAG model, facilitated through the integration of Langchain with Wikipedia. This endeavour serves not only as a testament to Langchain's capabilities but also as a foundational assessment, guiding our expectations and strategic planning for the project's advancement.

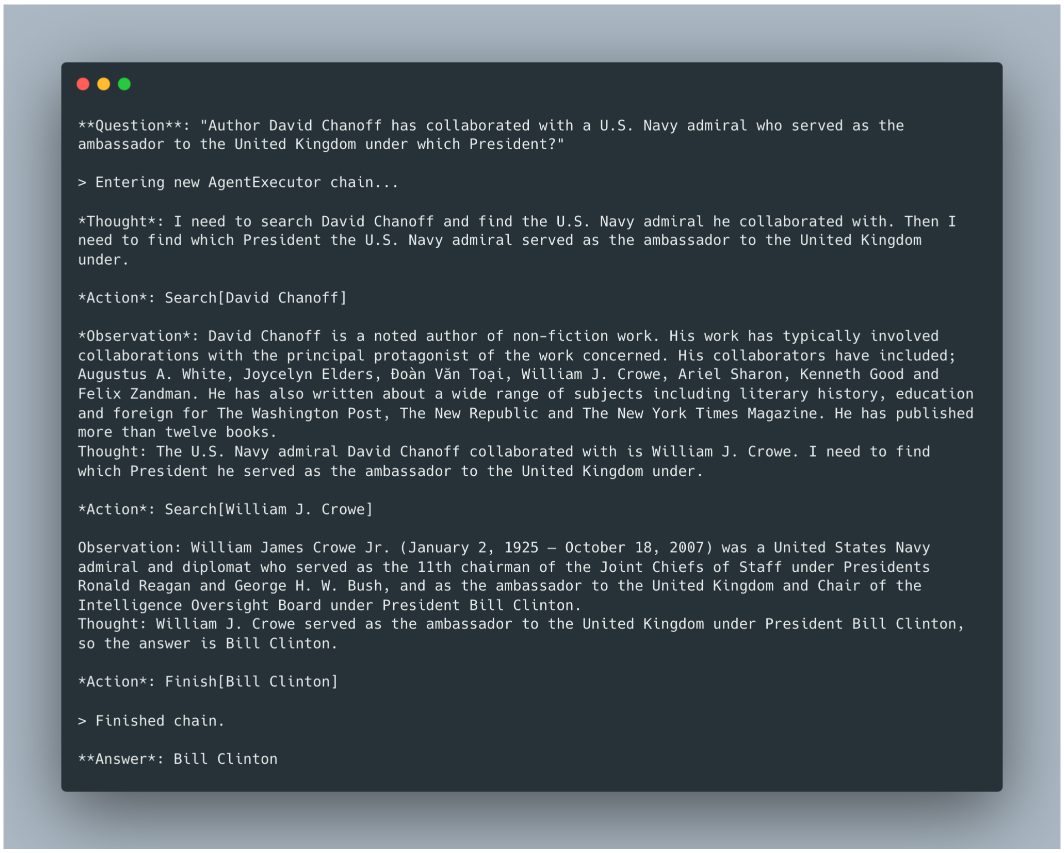

Figure 1: Langchain ReAct RAG output

3.2 LLM choice

After having a good idea of where we were heading, we decided to jump right in the project. At that point, we needed to think about which LLM use for the RAG, since it will change a lot of things regarding the different features of each of them. Thus, we started testing several LLMs such as GPT-4, Mistral, Zephyr and Llama2. We immediately understand that GPT-4 with OpenAI Python library would make the project a lot more easy, because it provides a function calling tool which is designed to give back a function from an initial set of giving function, along with the correct arguments. For example, with a search function that would take a query and search in a vector database, the project would just have been calling the library with a well defined function, then let the black box do the rest. Also, we really wanted to use open source tools, so we decided to use Llama2 with Ollama.

We performed various qualitative tests on this topic, and it was not easy. In a first place, how to know which output is better if the answer is correct for both? Then, do they all achieve correct results as consistently? Our "benchmark" was not very sophisticated, since we quickly understood that the true performance of our RAG would be part of the prompt engineering and the context giving to the LLM at inference time. The reason we choose Llama2 was because it was one of the easiest to use for us, as well as performing the best in our few tests on the final task. We decided to use Ollama, which is a tool making it easy to run LLMs locally. In fact, we initially wanted to use the model from HuggingFace, but we couldn't get access to Meta's model directly from the transformers library, which made us switch to Ollama for the rest of the project.

3.3 PDF reader and database choice

In parallel to the choice and benchmark of the models, we build a simple tool to load a PDF in chunks using PyPDF2 library, and then save it to a vector database. Regarding the choice of the vector database, we did not stressed this process to far, since it would not have a big impact on the outcome of the project. Having worked with ChromaDB previously, we knew it was easy to set up and have it running in Python, so we went for this database, rather than Milvus or Pinecone for example.

The embedding was also a consideration we had to take in, since we didn't want to use OpenAi or non open source project. We decided to try Jina's through HuggingFace, which would make our life easier for the integration in the code. It was very easy to integrate the embedding in the code, and plug it to ChromaDB for the storage of the PDF file.

3.4 ReAct framework

The subsequent phase of our project represented a pivotal juncture, characterised not only by its centrality to our objectives but also by the unforeseen challenges it presented. Contrary to our initial expectations, which posited the coding of the project's core components as the most daunting task—owing to the anticipated complexity of the ReAct format—the actual difficulties lay elsewhere.

Our primary challenge emerged in achieving consistency within the LLM responses, ensuring they conformed to the stringent specifications mandated by the ReAct framework. Specifically, the framework required outputs in a "Function name: description" format, a stipulation that significantly compounded the complexity of our task. Despite the initial focus on coding challenges, it was the process of prompt engineering and the refinement of system prompts that ultimately proved to be the most arduous and time-consuming aspect of our work.

Extensive experimentation with various prompt formulations and rigorous prompt engineering efforts were undertaken to elicit the desired response format from the LLM. Achieving parse-able outputs across a broad spectrum of scenarios emerged as a critical, yet elusive, goal. This endeavour not only underscored the intricacies of interfacing with advanced language models but also highlighted the nuanced challenges of tailoring LLM outputs to fit specific framework requirements. The struggle to ensure the reliability and consistency of these outputs in diverse contexts ultimately became the project's primary challenge.

3.5 Wrap-up and usage

Our final tool successfully implements the ReAct framework, but is very limited by the overthinking of the underlying LLM. In fact, actions are decided well in a first place, but seem to diverge from the initial objective, mainly after searching in the database. The final RAG almost never finds an answer to the question, feeling a constant need to find deeper information.

Thus the final results are pretty disappointing, even if of the global architecture of the ReAct framework works well. An inevitable improvement would be to consider a function calling feature plugged on an LLM, like OpenAI proposes. Such a feature would avoid the model to diverge from the original question and give a straight answer. Also, it would allow for more functionalities within the framework, which is currently limited by hallucinations from the LLM on the actions to call.

To use our project, follow the README on the GitHub page. Ultimately, simply run Ollama locally and start the main.py file using Python.

4 Conclusion

This research embarked on an ambitious endeavour to advance the field of artificial intelligence by pioneering the integration of RAG models for the nuanced task of processing and understanding PDF documents, utilising the innovative ReAct framework. Our initial explorations, leveraging the Langchain library, laid the groundwork for this project, unveiling the potential for a ReAct-based RAG model to significantly enhance machine comprehension and response generation.

The development and implementation phases involved meticulous methods for text extraction from PDF documents and the computation of embeddings, culminating in the successful integration within the Chroma vector database. This foundational work facilitated the creation of a robust RAG model capable of operating independently of Langchain, a milestone that underscores our project's success in developing an autonomous agent from the ground up.

Our findings illuminate the critical role of prompt engineering in refining and optimising the performance of RAG models, a testament to the intricate interplay between model architecture and input manipulation in achieving desired outcomes. This project not only demonstrates the practical application and versatility of RAG models when combined with the ReAct framework but also contributes valuable insights into their capabilities and limitations in handling complex document formats.

Reflecting on the challenges encountered and the methodologies employed, this research underscores the dynamic and evolving nature of artificial intelligence, pushing the boundaries of what is possible with current technologies. The implications of our work extend beyond the immediate project, offering a framework and a reference point for future endeavours aiming to harness the power of RAG models in diverse applications.

In conclusion, our project, detailed on our GitHub repository, highlights the importance of foundational research, the challenges of innovation, and the potential for future advancements in this vibrant field of study.

References

Zhuolun He and Bei Yu. Large language models for eda: Future or mirage? 2024.

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. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

Yun-Da Tsai, Mingjie Liu, and Haoxing Ren. Rtlfixer: Automatically fixing rtl syntax errors with large language models, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.